FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Cinemática Composta de Manipuladores Móveis

Gonçalo Daniel Ribeiro da Silva

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Orientador: Prof. Dr. Pedro Luís Cerqueira Gomes da Costa

Orientador INESC-TEC: Prof. Dr. Luís André Freitas da Rocha

Coorientador: Prof. Dr. José Luís Sousa de Magalhães Lima

21 de Julho de 2018

© Gonçalo Silva, 2018

Resumo

Um manipulador móvel resulta da integração de um manipulador com uma plataforma móvel, combinando a capacidade de manipulação do primeiro com a de locomoção do segundo. Desta forma, esta dissertação teve como objetivo a integração de ambas as cadeias cinemáticas de modo a se poder realizar movimentos com o manipulador enquanto a plataforma se move. Ao longo da sua execução foi usado o manipulador UR5 (constituído por 6 juntas rotativas) e uma plataforma omnidirecional de 4 rodas.

Para alcançar o objetivo pretendido foram utilizadas duas abordagens. Na primeira, o controlo da plataforma móvel e do manipulador é feito de forma separada, ou seja, os dois subsistemas presentes são independentes um do outro. Na segunda abordagem é feita a integração destes dois subsistemas, resultando, a partir desta, um sistema composto que pode ser interpretado como um manipulador (constituído por 9 juntas), o qual modela os movimentos da plataforma e do UR5. A resolução da cinemática deste manipulador é equivalente à resolução da cinemática de todo o sistema. Como forma de testar ambas as abordagens, definiram-se duas trajetórias para a ferramenta de trabalho. A comparação dos resultados mostrou que, na segunda abordagem, as trajetórias efetuadas pela mesma foram mais suaves. Além disso, a manipulabilidade ao longo das trajetórias foi maior do que na primeira abordagem.

A presente dissertação teve ainda como objetivo o desenvolvimento de um planeador de trajetórias capaz de gerar o caminho ótimo entre dois pontos, evitando colisões. Assim sendo, foi desenvolvido e implementado um planeador de trajetórias para o sistema, baseado no algoritmo A*. A metodologia implementada dividiu-se em duas fases. Primeiramente, uma vez definidos um ponto de início e destino, é gerada uma trajetória para o manipulador. Posteriormente, gera-se uma outra para a plataforma que garanta que a trajetória do manipulador esteja sempre ao seu alcance. Como forma de teste, definiram-se um ponto de início e destino para o manipulador e colocaram-se alguns obstáculos no ambiente de simulação. Para os testes realizados foram conseguidos bons resultados, visto que foi possível, ao manipulador, alcançar o ponto de destino, evitando colisões do sistema com os obstáculos.

Abstract

A mobile manipulator results from the integration of a manipulator with a mobile platform, combining the manipulation ability of the first with locomotion of the second. The main goal of this dissertation is combine both kinematics chains in a way that manipulator can perform movements while the platform moves. In this dissertation the UR5 manipulator (which has 6 rotational joints) and a 4-wheel omnidirectional platform are used.

Two approaches were used to solve this problem. In the first, the control of the mobile platform and the manipulator is done separately, so the two subsystems are independent of each other. The second approach integrates these two subsystems, resulting in a composite system that can be seen as a manipulator (which has 9 joints) that models the movements of the platform and the UR5. Solving the kinematics of this manipulator is equivalent to solving the kinematics of the whole system. As a way of testing both approaches, two trajectories for the end-effector were defined. The comparison of the results showed that in the second approach the trajectories performed by the end-effector were smoother. In addition, the manipulability along the trajectories were greater than in the first approach.

The present dissertation also has as objective the development of a trajectory planner capable of generating the optimal path between two points, avoiding collisions. Therefore, a trajectory planner for the system based on the A* algorithm was developed and implemented. The methodology implemented was divided into two phases. Firstly, a start point and a destination point are defined, then a trajectory is generated for the manipulator. Subsequently, a trajectory is generated for the platform that ensures that the trajectory of the manipulator is always within its reach. To test, a start and destination point were defined for the manipulator and some obstacles were placed in the simulation environment. The tests showed good results, as the manipulator reached the destination point avoiding collisions with obstacles.

Agradecimentos

Inicialmente quero agradecer aos meus orientadores, Prof. Dr. Pedro Costa, Prof. Dr. Luís Rocha e Prof. Dr. José Lima pelo conhecimento e apoio que me transmitiram ao longo da execução deste trabalho.

Quero também agradecer ao Prof. Dr. António Paulo Moreira por me dar a oportunidade de realizar esta dissertação no INESC-TEC.

Aos meus amigos ao longo desta jornada, Emanuel Pereira, Henrique Reis, Pedro Guedes, Pedro Moura e Sérgio Pinto. Obrigado por todas as nossas pausas para o café, foram nesses momentos de descontração que surgiram as soluções para muitos problemas. Emanuel, obrigado pelas boleias e pelos momentos de diversão. Pedro "Casilhas", obrigado por partilhares o teu conhecimento comigo e por todas as dicas certas no momento certo. Pedro Moura, obrigado pela ajuda e por todas as revisões de português. Sérgio, obrigado por estares sempre pronto a ajudar-me e pela companhia em muitos jantares ao longo destes meses.

Um agradecimento também ao Henrique, por todas as horas de conversa em que me ensinaste mais sobre o Brasil e eu te ensinei mais sobre Portugal. Não há nada melhor do que uma boa história para esquecer a pressão do dia a dia. Mostraste ser um grande amigo.

Aos meus pais e irmãos, um grande obrigado pelo apoio e incentivo ao longo deste percurso. Sem eles, nada disto era possível.

Filipa Magalhães, um obrigado especial para ti por todo o incentivo, ajuda e apoio que me deste durante todo o meu percurso académico, nos bons e nos maus momentos. Obrigado por toda a paciência que tiveste comigo, nem sempre foi fácil, eu sei. Sem ti, nunca teria chegado até aqui. Nunca esquecerei.

Um muito obrigado a todos!

Gonçalo Silva

"Projects we have completed demonstrate what we know.
Future projects decide what we will learn"

Mohsin Tiwana

Conteúdo

1 Introdução **1**
1.1 Contexto . . . 1
1.2 Objetivos . . . 2

2 Revisão bibliográfica **3**
2.1 Manipuladores . . . 4
2.1.1 Tipos de Manipuladores . . . 4
2.1.1.1 Robô Cartesiano (PPP) . . . 5
2.1.1.2 Robô Cilíndrico (RPP) . . . 6
2.1.1.3 Robô Esférico (RRP) . . . 6
2.1.1.4 Robô *Scara* (RRP) . . . 7
2.1.1.5 Robô Antropomórfico (RRR) . . . 7
2.1.2 Manipulador UR5 . . . 7
2.2 Robôs Móveis . . . 8
2.2.1 Disposição das rodas . . . 9
2.2.1.1 Configuração Diferencial . . . 10
2.2.1.2 Configuração *Synchro* . . . 11
2.2.1.3 Configuração Triciclo . . . 11
2.2.1.4 Configuração Omnidirecional . . . 12
2.2.2 Manipuladores móveis . . . 13
2.2.2.1 Planeamento de trajetórias . . . 13
2.2.2.2 Estratégias de controlo coordenado . . . 15

3 Cinemática de um manipulador industrial e de uma plataforma omnidirecional **19**
3.1 Cinemática direta do manipulador UR5 . . . 20
3.2 Cinemática inversa do manipulador UR5 . . . 22
3.3 Jacobiano . . . 28
3.3.1 Manipulabilidade . . . 28
3.4 Cinemática do veículo omnidirecional . . . 29

4 Cinemática composta de um manipulador móvel **33**
4.1 Ferramentas . . . 33
4.1.1 *Virtual Robot Experimentation Platform* (V-REP) . . . 33
4.1.2 *Robot Operation System* (ROS) . . . 34
4.2 Algoritmo de navegação da plataforma móvel . . . 37
4.3 Controlo do manipulador . . . 40
4.4 Controlo baseado em dois sistemas independentes . . . 44
4.5 Controlo do sistema integrado . . . 44

4.5.1 Biblioteca *TRAC-IK* 46
4.6 Resultados 50
4.7 Conclusão 57

5 Planeamento de trajetórias **59**
5.1 Algoritmo A* 60
5.1.1 Aplicação na plataforma e no manipulador 63
5.2 Implementação 65
5.2.1 Algoritmo implementado 68
5.2.2 Integração ROS 69
5.3 Resultados 69
5.4 Conclusão 77

6 Conclusões e propostas de trabalho futuro **79**
6.1 Trabalho futuro 80

Lista de Figuras

1.1 Stock estimado de robôs operacionais entre 2008-2016 e evolução esperada entre 2017-2020 [1]. . . . 2

2.1 Exemplo de Manipulador Móvel [8]. . . . 4
2.2 Junta rotativa à esquerda e prismática à direita. . . . 5
2.3 Robô cartesiano da *Epson* [11]. . . . 5
2.4 Robô cilíndrico [11]. . . . 6
2.5 Robô esférico [11]. . . . 6
2.6 Robô *scara* [11]. . . . 7
2.7 Robô antropomórfico [11]. . . . 7
2.8 Manipulador robótico UR5 [14]. . . . 8
2.9 Tipos de locomoção dos robôs móveis. . . . 9
2.10 Tipos de rodas [16]. . . . 10
2.11 Possibilidades de locomoção configuração diferencial [16]. . . . 10
2.12 Configuração *Synchro* [3]. . . . 11
2.13 Configuração triciclo. . . . 11
2.14 Configuração omnidirecional [16]. . . . 12
2.15 Movimentos de uma plataforma omnidirecional [17]. . . . 12

3.1 Sistema de coordenadas segundo o método DH [32]. . . . 21
3.2 Cálculo de P_5^0 [32]. . . . 23
3.3 Vista superior para o cálculo de θ_1 [32]. . . . 23
3.4 Vista superior para o cálculo de θ_5 [32]. . . . 24
3.5 Cálculo de θ_6 [32]. . . . 25
3.6 Plano que contém referencial 1, 2 e 3 [32]. . . . 25
3.7 Conjunto de configurações do manipulador UR5. . . . 27
3.8 Configuração omnidirecional com 4 rodas *mecanum*. . . . 29

4.1 Manipulador móvel. . . . 34
4.2 Funcionamento de um tópico. . . . 35
4.3 Funcionamento de um serviço. . . . 35
4.4 Comunicação com V-REP via ROS. . . . 36
4.5 Cálculo da velocidade em x e y. . . . 37
4.6 Evolução da posição e orientação da plataforma. . . . 39
4.7 Trajetória efetuada pela plataforma. . . . 39
4.8 Relações entre os referenciais do sistema. . . . 40
4.9 Plataforma e manipulador. . . . 41
4.10 Resultado da cinemática inversa. . . . 43
4.11 Diagrama de atividade para controlo dos dois subsistemas. . . . 44

4.12 Aproximação da plataforma omnidirecional por um manipulador. 45
4.13 Diagrama de atividade para controlo do sistema. 45
4.14 Especificação de um robô através de elos e juntas. 46
4.15 Desenvolvimento da junta prismática em x. 47
4.16 Desenvolvimento da junta prismática em y. 48
4.17 Desenvolvimento da junta rotacional. 48
4.18 Ligação da base do UR5 à plataforma móvel. 49
4.19 Posição inicial e final. 50
4.20 Ambiente da primeira simulação. 50
4.21 Resultados da primeira abordagem para trajetória retangular. 51
4.22 Resultados da segunda abordagem para trajetória retangular. 53
4.23 Ambiente da segunda simulação. 54
4.24 Resultados da primeira abordagem para trajetória sinusoidal. 55
4.25 Resultados da segunda abordagem para trajetória sinusoidal. 56

5.1 Função $f(n)$, $g(n)$ e $h(n)$. 60
5.2 Heurística distância de *Manhattan*. 61
5.3 Zona de Segurança. 63
5.4 Trajetória gerada para a plataforma móvel. Caso 1. 64
5.5 Trajetória gerada para a plataforma móvel. Caso 2. 65
5.6 Trajetória gerada para o manipulador. 65
5.7 Zona de segurança em torno da trajetória do manipulador. 66
5.8 Cálculo do ponto mais perto ao ponto P. 67
5.9 Diagrama de atividade para seguimento das trajetórias. 68
5.10 Arquitetura *ROS*. 69
5.11 Ambiente de simulação. 70
5.12 Trajetórias resultantes do A*. A azul a trajetória do manipulador e a vermelho a trajetória da plataforma. 70
5.13 Resultados do primeiro teste no simulador *V-REP*. 71
5.14 Comparação das trajetórias desejadas e efetuadas. 72
5.15 Novo ambiente de simulação. 73
5.16 Trajetórias resultantes do A* para o novo ambiente de simulação. A azul a trajetória do manipulador e a vermelho a trajetória da plataforma. 73
5.17 Resultados do segundo teste no simulador *V-REP*. 74
5.18 Comparação das trajetórias desejadas e efetuadas. 75
5.19 Representação do limite das juntas rotativas no círculo trigonométrico. 76
5.20 Evolução das coordenadas das juntas ao longo da simulação. 77

Lista de Tabelas

2.1 Especificações técnicas do manipulador UR5 [15]. 8

3.1 Parâmetros de DH para o robô UR5 21

4.1 Resultados da cinemática inversa. 42
4.2 Resultado para o ponto definido 49

Abreviaturas e Símbolos

CRIIS	Centre for Robotics in Industry and Intelligent Systems
KDL	Orocos Kinematics and Dynamics Library
PCB	Printed Circuit Board
PQS	Programação quadrática sequencial
ROS	Robot Operation System
SCARA	Selective Compliance Assembly Robot Arm
URDF	Unified Robot Description Format
V-REP	Virtual Robot Experimentation Platform
XML	Extensible Markup Language

Capítulo 1

Introdução

1.1 Contexto

A presente dissertação insere-se no projeto *ColRobot* que está a ser desenvolvido no CRIIS[1] (*Centre for Robotics in Industry and Intelligent Systems*) e pretende desenvolver um manipulador móvel capaz de colaborar com Humanos, mais especificamente, operadores na área da indústria automóvel e aeroespacial fornecendo-lhes ferramentas, *kits* ou segurando peças enquanto o operador as trabalha.

Uma vez que o robô terá a capacidade de navegar por todo o chão da fábrica, será possível ele mesmo poder recolher as ferramentas, preparar os *kits* e depois entregá-los ao operador. Esta capacidade de movimentação do robô permite aumentar a flexibilidade do processo de produção.

A tecnologia e a robótica são cada vez mais uma necessidade e uma constante na sociedade e no mundo. Com o passar dos anos tem-se observado um acréscimo de sistemas robóticos aplicados à indústria, sendo de esperar que este crescimento se mantenha nos próximos anos (figura 1.1). A implementação destes sistemas robóticos numa indústria melhora a produtividade e a flexibilidade das linhas de produção, tornando-a numa unidade inovadora e mais rentável, o que é crucial para manter a competitividade no mercado. O *ColRobot* é um exemplo prático de um desses sistemas pois combina, entre outras coisas, a habilidade do robô com as do Humano permitindo aos trabalhadores estarem mais focados nas atividades que geram valor.

Assim sendo, uma vez que o projeto *ColRobot* tem objetivos realmente interessantes que visam a melhoria e inovação, esta proposta de dissertação torna-se bastante aliciante.

[1] https://www.colrobot.eu/

Figura 1.1: Stock estimado de robôs operacionais entre 2008-2016 e evolução esperada entre 2017-2020 [1].

1.2 Objetivos

Um dos desafios atuais dos manipuladores móveis é o planeamento integrado da sua trajetória. Assim sendo esta dissertação terá como objetivo integrar a cadeia cinemática de um manipulador com a de um robô omnidirecional, permitindo desta forma que o manipulador possa realizar movimentos enquanto a plataforma se move. Além disso, esta dissertação visa o desenvolvimento e a integração de um planeador de trajetórias no sistema.

Deste modo, foi definido um plano de trabalhos dividido em duas fases:

Fase de Preparação

1. Estudo do estado da arte relativo ao cálculo de cadeias cinemáticas para manipuladores móveis;
2. Estudo de bibliotecas de suporte para cálculo de cinemática inversa e respetivo planeamento de trajetórias;
3. Estudo de diferentes ambientes de simulação.

Fase de Desenvolvimento

1. Desenho da cadeia cinemática composta de um manipulador móvel;
2. Desenvolvimento/Integração de um planeador de trajetórias;
3. Teste e validação do sistema desenvolvido;
4. Conclusão e escrita da tese.

Capítulo 2

Revisão bibliográfica

O desenvolvimento do primeiro robô industrial (*Unimate*) remonta a 1961, ano em que *George Devol* e *Joseph Engelberger* (fundadores da empresa *Unimation*) desenvolveram o primeiro protótipo oficial de um robô industrial, que foi instalado na fábrica da *General Motors* em *Trenton* [2]. Obviamente este primeiro manipulador era bastante simples quando comparado com os que existem atualmente. Características como velocidade, precisão e segurança foram melhorando até aos dias de hoje fazendo destes manipuladores um caso de sucesso devido ao bom desempenho que demonstram em tarefas repetitivas que seriam enfadonhas e desmotivadores para os operadores. Apesar de tudo, este tipo de manipuladores apresenta uma grande limitação: falta de mobilidade [3].

Para colmatar esta lacuna o manipulador foi integrado numa plataforma móvel surgindo assim os "manipuladores móveis", denominados desta forma devido à capacidade de navegarem livremente num determinado ambiente de trabalho sem intervenção humana. Desta forma, ao aliarem a locomoção com a manipulação aumentam significativamente a sua área de trabalho comparativamente com os manipuladores fixos. Além disso, graças à capacidade de locomoção os manipuladores móveis podem ser usados em diversas áreas, tais como segurança (robôs de vigilância) ou doméstica (robôs de limpeza). Na área da indústria os manipuladores móveis têm vindo aos poucos a ocupar um papel preponderante na gestão dos supermercados logísticos, nomeadamente na execução de tarefas de *Kitting* de componentes e respetivo transporte para a linha de produção, libertando os operadores humanos para tarefas mais nobres e de maior valor acrescentado para o produto final [4]. Ao serem integrados com o sistema de rastreabilidade de componentes, permitem ainda uma redução de falhas no cálculo do inventário (evitando a rutura de stock de componentes para produção) e um controlo mais apertado dos produtos acabados na linha de produção. Adicionalmente, a sua aplicação em tarefas de assemblagem colaborativa com operadores humanos, em linhas de produção altamente flexíveis e adaptáveis (*Factories of the Future*) tem vindo cada vez mais a ser explorada, sobretudo em projetos H2020 (exemplos: *ColRobot* [5], *ScalABLE 4.0* [6], *ILIAD* [7] entre outros).

Os manipuladores móveis podem ter diferentes tipos de tração e ser constituídos por diferentes tipos de manipuladores, assim sendo ao longo deste capítulo irão ser abordadas as diferentes

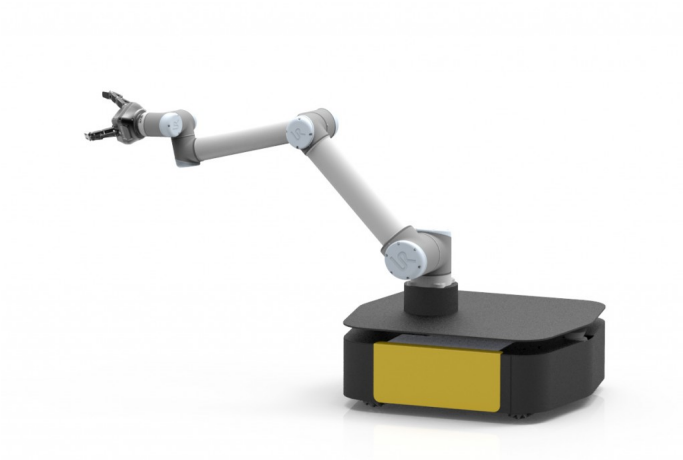


Figura 2.1: Exemplo de Manipulador Móvel [8].

possibilidades para cada um dos casos, tendo em vista o desenvolvimento da cinemática composta do manipulador.

Dado que o planeamento de trajetórias é também uma das fases principais do desenvolvimento de um manipulador móvel, pois é o que permite ao manipulador navegar autonomamente por todo o seu ambiente de trabalho, neste capítulo serão abordados diferentes métodos de planeamento de trajetória e heurísticas para o controlo da mesma.

2.1 Manipuladores

Um manipulador é constituído por um conjunto de blocos rígidos denominados elos (*links*) que são ligados através de juntas (*joints*). O último elo está ligado ao punho do manipulador, ou seja, à ferramenta de trabalho. Normalmente os manipuladores industriais são constituídos por 6 juntas, em que as 3 primeiras (juntas primárias) são responsáveis por posicionar a estrutura e as restantes por orientar a ferramenta de trabalho [9].

Uma das principais características de um manipulador móvel é o seu número de graus de liberdade, pois esta característica determina a quantidade e o tipo de movimentos que o manipulador será capaz de executar [10].

Os dois principais tipos de juntas que existem são as:

- Rotativas (R) onde o movimento relativo dos elos é rotacional.
- Prismáticas (P) onde o movimento relativo dos elos é linear.

Tanto as juntas rotativas como as prismáticas possuem 1 grau de liberdade.

2.1.1 Tipos de Manipuladores

O volume de trabalho de um manipulador é definido pela região dentro da qual o robô consegue posicionar a sua ferramenta de trabalho e está relacionado com a sua estrutura cinemática que é

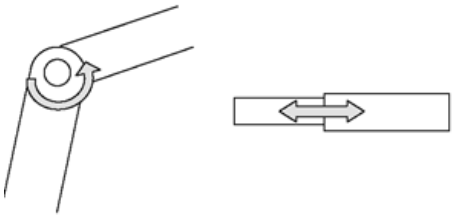


Figura 2.2: Junta rotativa à esquerda e prismática à direita.

dada pela configuração das suas juntas primárias. Para se simplificarem os cálculos do volume de trabalho de cada manipulador, considera-se que as juntas rotacionais varrem 360° e que as prismáticas têm comprimento de L. Existem várias combinações possíveis de juntas prismáticas e rotativas para formar a estrutura cinemática, contudo na prática as mais usadas são:

- Cartesiana (PPP).
- Cilíndrica (RPP).
- Esférica (RRP).
- Articulado horizontal ou *Scara* (RRP).
- Articulado vertical ou Antropomórfico (RRR).

2.1.1.1 Robô Cartesiano (PPP)

Consegue alcançar qualquer posição de um cubo de lado L:

$$VolumeTrabalho = L \times L \times L \tag{2.1}$$

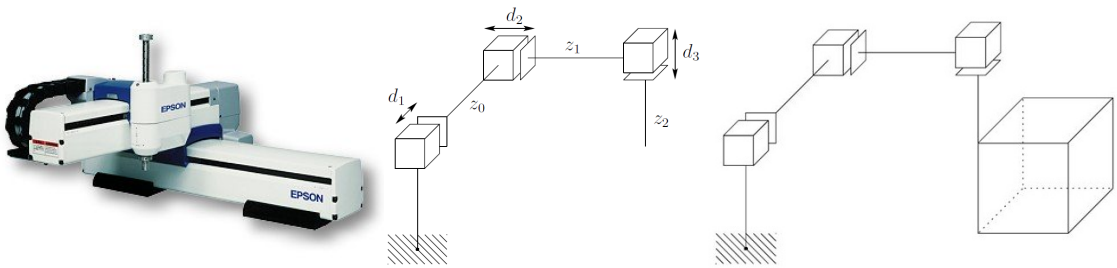


Figura 2.3: Robô cartesiano da *Epson* [11].

2.1.1.2 Robô Cilíndrico (RPP)

Consegue alcançar qualquer ponto no interior de um cilindro de altura L e raio 2L, exceto os pontos do cilindro interior de altura L e raio L:

$$VolumeTrabalho = 9,42 \times L \times L \times L \tag{2.2}$$

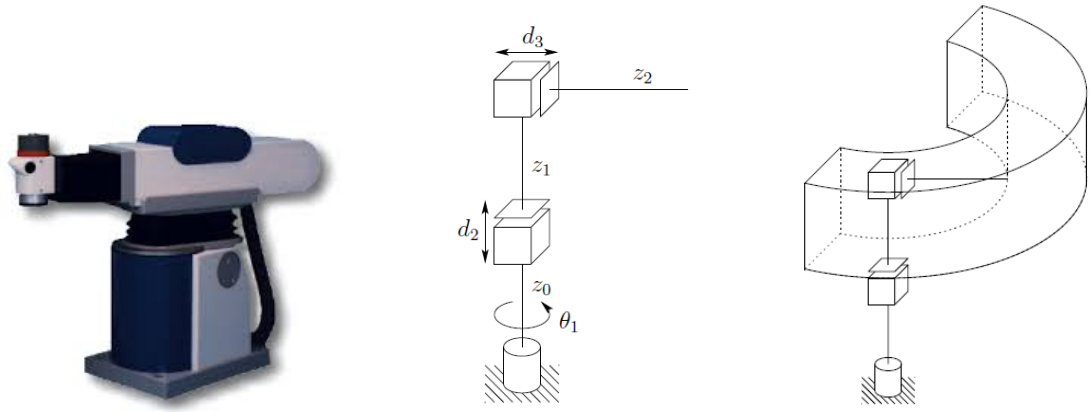


Figura 2.4: Robô cilíndrico [11].

2.1.1.3 Robô Esférico (RRP)

Consegue alcançar qualquer ponto de uma esfera de raio 2L, exceto a esfera interna de raio L:

$$VolumeTrabalho = 29,32 \times L \times L \times L \tag{2.3}$$

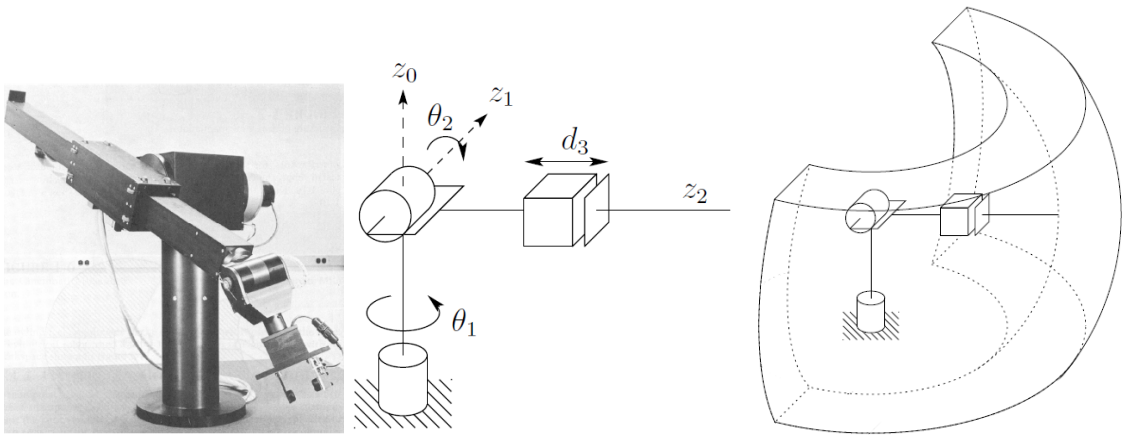


Figura 2.5: Robô esférico [11].

2.1.1.4 Robô *Scara* (RRP)

Consegue alcançar qualquer ponto de um cilindro de raio 2L e altura L:

$$VolumeTrabalho = 12,56 \times L \times L \times L \tag{2.4}$$

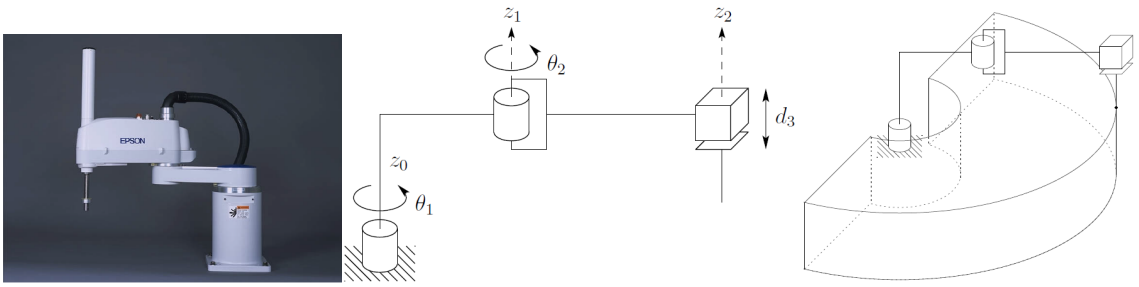


Figura 2.6: Robô *scara* [11].

2.1.1.5 Robô Antropomórfico (RRR)

Consegue alcançar qualquer ponto de uma esfera de raio 2L:

$$VolumeTrabalho = 33,51 \times L \times L \times L \tag{2.5}$$

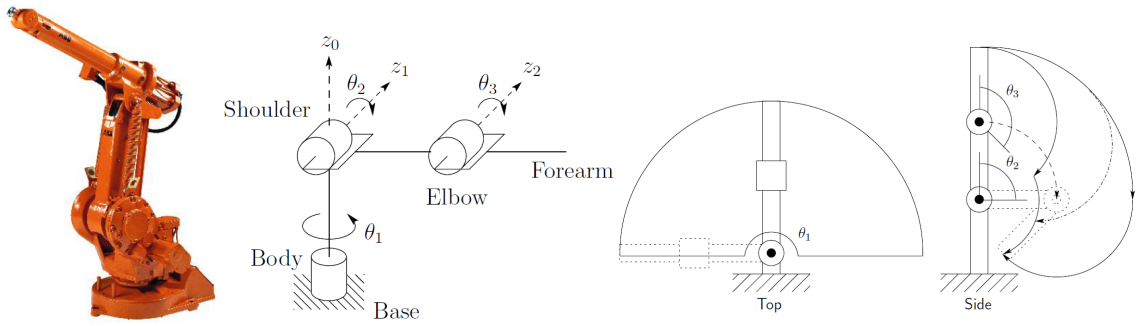


Figura 2.7: Robô antropomórfico [11].

2.1.2 Manipulador UR5

O manipulador escolhido para realizar testes nesta dissertação foi o UR5, apresentado na figura 2.8, fabricado pela *Universal Robotics*. Trata-se de um manipulador colaborativo leve, rápido, seguro e fácil de programar muito apreciado por investigadores e utilizado na indústria em tarefas de *pick-and-place*. A segurança é uma das características principais do UR5, o que faz dele ideal para trabalhar em conjunto com humanos. Devido ao seu mecanismo de segurança, é possível

definir um limite de força aplicada ao manipulador, de forma a que quando um humano entrar em contacto com o robô e caso seja aplicada uma força superior a esse limite, o manipulador para imediatamente de funcionar [12]. Graças à sua flexibilidade, estes manipuladores têm sido utilizados em diversas indústrias para desempenhar diferentes funções. Na indústria automóvel é comum utilizar-se este manipulador para tarefas de *packing*, montagem, *pick-and-place*, colagem e inspeção de qualidade. Várias empresas utilizam este manipulador, como é o caso da *Continental* que usa vários exemplares para produção e manipulação de placas PCB, reduzindo em até 50% o tempo gasto nestas tarefas, a *Lear* que implementou estes manipuladores para controlo de qualidade e a *Nissan Motor Company* que os utilizou para tarefas de montagem [13].

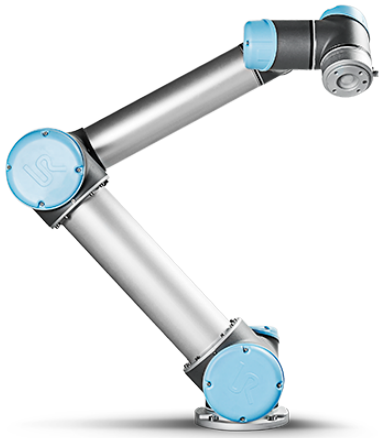


Figura 2.8: Manipulador robótico UR5 [14].

Tabela 2.1: Especificações técnicas do manipulador UR5 [15].

Modelo	UR5
Peso	18,4 kg
Alcance	850 mm
Espaço ocupado	Ø149mm
Graus de mobilidade	6 juntas rotativas
Gama de trabalho	$\pm$360 º (todas as articulações)
Velocidade máxima articulações	$\pm$180 º/s
Velocidade máxima ferramenta	1 m/s
Repetibilidade	$\pm$0,1 mm
Temperatura	0-50 ºC
Consumo de energia	Min. 90 W, Normal 150 W, Máx. 325 W

2.2 Robôs Móveis

Um robô móvel tem a capacidade de se mover de um lugar para outro sem intervenção humana, ou seja, autonomamente [16]. Devido à sua locomoção não se encontram limitados a um espaço de trabalho fixo, podendo mover-se livremente.

O tipo de ambiente em que está inserido o robô afeta diretamente o tipo de locomoção escolhido para o mesmo. Assim, existem vários tipos de locomoção para cada tipo de ambiente, como se pode ver na figura 2.9. No âmbito desta dissertação, abordam-se somente os robôs móveis ter-

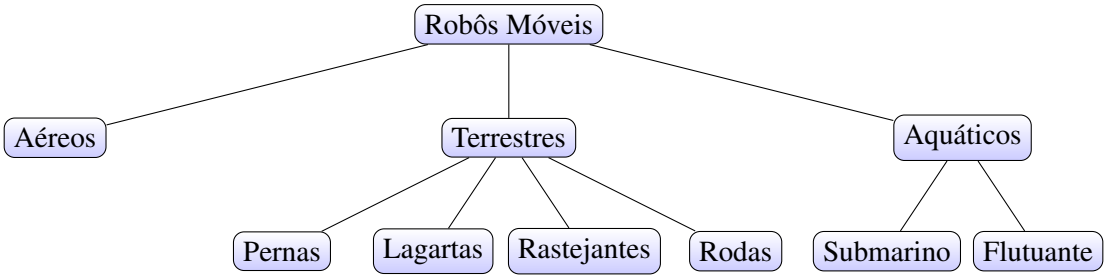


Figura 2.9: Tipos de locomoção dos robôs móveis.

restres com rodas pois além deste mecanismo de locomoção ser um dos mais populares na área robótica móvel apresenta baixa complexidade mecânica e um baixo consumo energético [3].

A mobilidade do robô é ainda afetada pelo tipo de roda e pela sua disposição na estrutura do robô. Existem basicamente duas classes de rodas: a convencional e a especial, que se dividem nos seguintes tipos (figura 2.10):

- Convencionais
  - Roda Fixa: são as mais comuns usadas em robótica. O eixo de rotação está fixo à estrutura do robô e são normalmente associadas à tração do robô.
  - Roda Castor: é orientável conseguindo rodar livremente sobre um eixo perpendicular ao seu eixo de rotação. Funciona sobretudo para dar estabilidade como roda de direção.
- Especiais
  - Roda Universal: além da rotação normal na direção do plano da roda, graças aos rolamentos montados na sua superfície também consegue rolar na direção do seu eixo.
  - Roda Mecanum: similar à roda Universal, contudo os rolamentos estão montados com ângulo diferente de 90 °, geralmente $\pm$ 45 °

2.2.1 Disposição das rodas

Existem várias combinações para os diferentes tipos de rodas, que levam à construção de diferentes robôs móveis. De seguida analisam-se as seguintes configurações de robôs móveis:

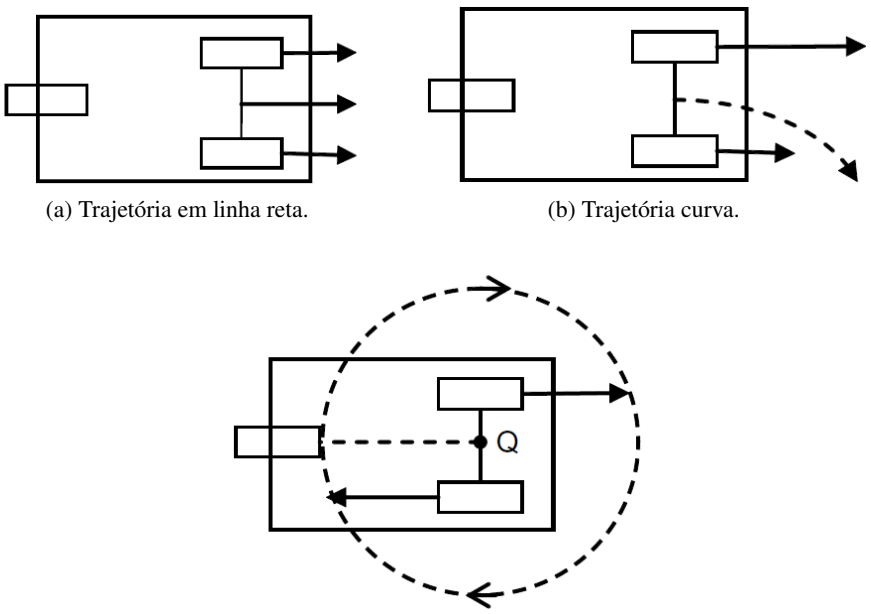
- Configuração diferencial
- Configuração *synchro*
- Configuração triciclo
- Configuração omnidirecional

(a) Roda fixa. (b) Roda Castor. (c) Roda Universal. (d) Roda Mecanum.

Figura 2.10: Tipos de rodas [16].

2.2.1.1 Configuração Diferencial

Esta configuração consiste em duas rodas montadas na estrutura segundo o mesmo eixo, alimentadas de forma independente por dois motores. A estabilidade da estrutura é garantida por uma (ou mais) roda Castor. Esta configuração não permite que o robô se movimente de forma perpendicular ao eixo das rodas, sendo necessário fazer uma rotação para alinhar com o ponto de destino. Essa rotação é conseguida através da velocidade de rotação de cada roda. Se ambas as rodas se movimentarem com a mesma velocidade, o robô move-se em linha reta. Se uma das rodas rodar com maior velocidade que a outra, o robô irá seguir uma trajetória curva. Se ambas rodarem com a mesma velocidade e sentidos opostos, o robô gira em torno do ponto central do eixo formado entre as duas rodas [16].



(a) Trajetória em linha reta. (b) Trajetória curva.

(c) Trajetória circular em torno de Q.

Figura 2.11: Possibilidades de locomoção configuração diferencial [16].

2.2.1.2 Configuração *Synchro*

Esta é uma configuração interessante porque embora seja composto por 3 rodas, necessita de apenas 2 motores. Um dos motores é responsável pela rotação das rodas que produz movimento enquanto o outro é responsável pela orientação das mesmas. As rodas estão acopladas mecanicamente de tal forma que rodam sempre à mesma velocidade e na mesma direção.

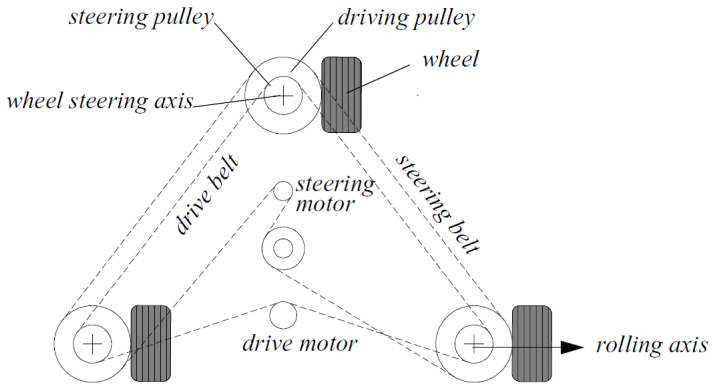


Figura 2.12: Configuração *Synchro* [3].

2.2.1.3 Configuração Triciclo

A configuração triciclo é muito usada atualmente na indústria para transporte de cargas pesadas. Esta é uma topologia que apresenta apenas uma roda responsável pela tração e direção. Para estabilidade, possui duas rodas livres na parte traseira de modo a ter sempre 3 pontos de contacto.

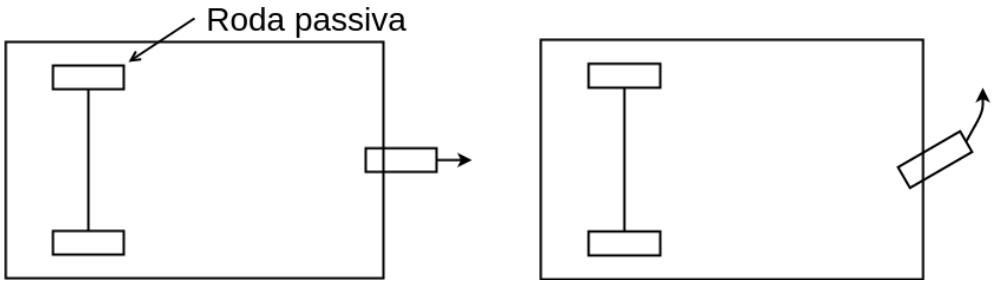


Figura 2.13: Configuração triciclo.

Para andar em linha reta, a roda de tração deve ser colocada de acordo com a figura 2.13a, por outro lado, caso se queira efetuar uma trajetória curva, esta deve ser alinhada com a direção desejada (figura 2.13b). Se o eixo da roda frontal estiver com uma orientação de 90^{o} em relação ao eixo das rodas traseiras, o robô irá efetuar uma trajetória circular cujo centro estará situado no ponto médio entre as duas rodas traseiras.

2.2.1.4 Configuração Omnidirecional

Os robôs omnidirecionais têm máxima manobrabilidade uma vez que conseguem mover-se em qualquer direção do plano sem necessidade de se reorientarem, algo que as plataformas com rodas convencionais não são capazes de fazer. Esta característica torna mais atrativa a sua utilização em ambientes apertados quando comparado com outras configurações, além de que se evita manobras complexas para se atingir a posição pretendida. Na figura 2.14 pode ver-se uma das configurações omnidirecionais mais utilizadas.

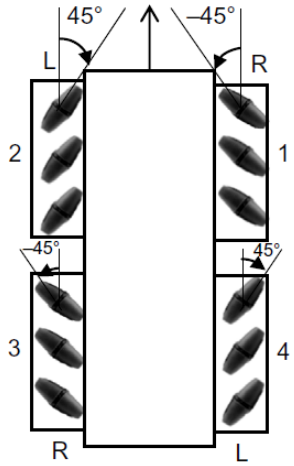


Figura 2.14: Configuração omnidirecional [16].

Esta configuração é composta por 4 rodas *mecanum* em que os rolos estão orientados a 45°. Através da variação da velocidade de cada roda, o robô consegue mover-se ao longo de qualquer trajetória no plano bidimensional. Na figura 2.15 pode-se perceber o sentido dos movimentos de uma plataforma omnidirecional de acordo com o sentido do movimento das rodas. Uma das apli-

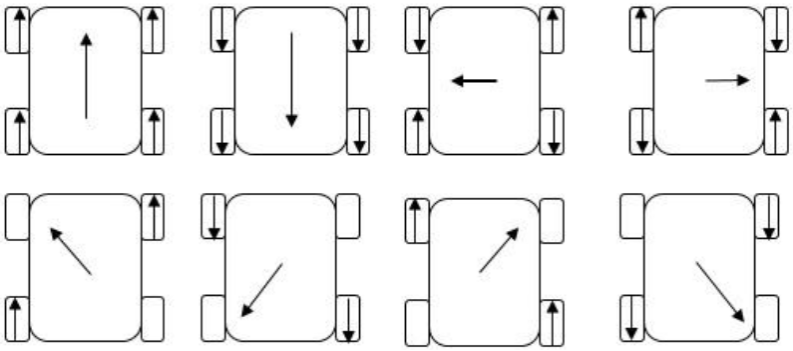


Figura 2.15: Movimentos de uma plataforma omnidirecional [17].

cações em que a configuração omnidirecional é especialmente vantajosa, é exatamente na manipulação móvel. Em tais situações é ideal que a plataforma consiga mover-se sem grande impacto na posição do manipulador. A configuração omnidirecional consegue oferecer tais capacidades [3].

Devido a estas vantagens, a plataforma usada nesta dissertação para dar mobilidade ao manipulador será omnidirecional.

2.2.2 Manipuladores móveis

O espaço de trabalho de um manipulador é uma das características fundamentais de um manipulador. Deve-se ter em atenção que os manipuladores não devem operar em zonas onde se atinja singularidades, pois torna difícil ou impossível o controlo do mesmo. Como já foi referido anteriormente, a integração de um manipulador com uma plataforma móvel aumenta significativamente a área de trabalho do manipulador, tornando-o cinematicamente redundante. Apesar das inúmeras vantagens oferecidas pelos manipuladores móveis, é preciso ter em conta que o planeamento da trajetória e o controlo do sistema são assuntos mais desafiantes, quando comparados com os manipuladores fixos. Considerando que se pretende que a ferramenta de trabalho alcance um determinado ponto no referencial global, é necessário entender como coordenar o movimento da plataforma e do manipulador, ou seja, trata-se de resolver um problema de coordenar locomoção com manipulação.

Para que o manipulador móvel possa navegar de forma segura deve-se dividir o problema em duas etapas:

1. Planeamento da trajetória: dado o ponto inicial e ponto final é definido um caminho que deve ser seguido pelo manipulador para se movimentar entre esses dois pontos. Um caminho é constituído por uma sequência de posições ao longo do espaço de trabalho.

2. Seguimento da trajetória: tenta-se posicionar o manipulador sobre o caminho previamente definido levando o erro da trajetória para zero, através da atuação das variáveis de controlo do manipulador móvel.

2.2.2.1 Planeamento de trajetórias

O planeamento de uma trajetória tem como objetivo encontrar o caminho ótimo entre dois pontos, que geralmente representa o caminho mínimo entre estes, evitando colisões. Para tal, uma das técnicas baseia-se na construção de um grafo em função do mapa e posterior pesquisa do caminho mínimo no mesmo. Existem várias abordagens para gerar o grafo a ser pesquisado, duas das mais conhecidas são:

- *Roadmap*: o mapa é representado por um conjunto de nós que podem significar uma localização, interligados por um conjunto de retas que podem significar um caminho físico entre esses nós.

- Decomposição em células: o mapa é dividido num conjunto de células e efetua-se a ligação de cada célula às células vizinhas.

Tendo o grafo pode então efetuar-se a pesquisa para descobrir a melhor trajetória. As soluções mais diretas para resolver o problema, baseiam-se em algoritmos de pesquisa exaustiva, assim denominados uma vez que seguem uma ordem pré-definida e iterativa de procura até atingirem o destino. Estes algoritmos são chamados de algoritmos sem informação, pois não conhecem o grafo no início da execução. Dentro desta gama de algoritmos os mais conhecidos são [18]:

- Pesquisa em profundidade: em cada nó explora a primeira ligação que encontrar até chegar a um nó sem ligações, só depois recua.
- Pesquisa em largura: explora todas as ligações de um nó antes de passar ao nó seguinte.
- Pesquisa em profundidade limitada: idêntico à pesquisa em profundidade, contudo existe um limite de profundidade que faz o algoritmo retornar mesmo que o nó possua ligações.
- Pesquisa em aprofundamento iterativo: idêntico à pesquisa em profundidade limitada, contudo a limite vai sendo aumentando até encontrar o destino.

Estes algoritmos, apesar de completos (isto é, encontram solução sempre que esta existe), apresentam a lacuna de terem um tempo de execução muito longo e consumirem muita memória. Para colmatar esta lacuna surgiram os algoritmos com heurística, assim denominados pois usam informação que vão recolhendo ao longo da pesquisa para determinarem os próximos passos, diminuindo assim o tempo de execução do programa. Dentro desta categoria, os algoritmos mais reconhecidos são:

- Algoritmo *Dijkstra*: este algoritmo baseia-se no custo de cada movimento. Partindo da posição inicial, o algoritmo marca todos os vizinhos com o custo deslocando-se depois sempre para o vizinho com o custo mais curto. Desta forma em cada iteração vai explorar sempre o nó mais perto do nó inicial. Este algoritmo efetua muitos cálculos e todos os nós são guardados em memória, portanto este algoritmo também não é o ideal.
- Algoritmo A*: pode ser visto como uma melhoria do *Dijkstra*, pois usa uma heurística de custo para determinar a ordem em que os nós são processados. Para escolher qual o melhor caminho, o algoritmo além de usar a informação de quais nós estão mais perto, também entra com a informação da distância que falta para alcançar o destino.

O algoritmo *Dijkstra* apesar de ser completo, efetua muitos cálculos e guarda todos os nós em memória o que faz com que também não seja um algoritmo ideal. Já o A* também se trata de um algoritmo completo e é o algoritmo mais eficiente entre todos os aqui apresentados.

Os algoritmos até agora abordados são apropriados para problemas em que o grafo é estático. Quando se trata de problemas em tempo real, em que o espaço de pesquisa está em constante alteração, estes têm de refazer o planeamento desde o início. Por esse motivo, começaram a ser desenvolvidos variantes do A* para serem implementadas em tempo real. Estas variantes foram desenvolvidas com o objetivo de não ter de se refazer o planeamento desde o início, sendo apenas necessário ajustar a trajetória. Dentro desses algoritmos os mais conhecidos são o IDA*, ARA*, MA*, SMA*, D*, *Focused* D*, D* *lite* entre outros.

Estes algoritmos ou geram uma solução ótima, tal como o A*, ou então geram uma solução subótima, ou seja, uma solução pior. As vantagens destes algoritmos são que o MA* e o SMA* usam menos memória, o IDA* e o ARA* executam em menos tempo, enquanto os restantes têm a vantagem de em alguns casos em tempo real executarem em menos tempo que o A* [18].

Segundo [19], algoritmo IDA* consiste em definir-se um limite k para a função heurística *f*, da mesma forma que o algoritmo de pesquisa em aprofundamento iterativo. Este algoritmo executa o A*, contudo não são expandidos os nós que tenham f(n)>k. Caso o algoritmo não encontre o destino, incrementa-se o valor de k e executa-se novamente o A*. Este procedimento repete-se até se encontrar o destino ou até se esgotar um tempo predefinido. Caso o algoritmo não encontre o destino, a solução será a que estiver mais perto deste.

O algoritmo ARA* tem a capacidade de construir soluções subótimas em tempos reduzidos. Usando a heurística *weighted* consegue-se reduzir o tempo de execução do A*. Desta forma, o ARA* encontra uma solução subótima e vai melhorando enquanto houver tempo disponível [20].

O algoritmo MA* surgiu em 1989 [21]. Este consiste em retirar da árvore os nós menos promissores (ou seja, com maior custo), quando a memória está cheia, para abrir espaço para novos nós. Para tal, quando a lista aberta e a fechada chegam a um limite predefinido de nós, o próximo nó a entrar na lista aberta faz com que o menos promissor seja retirado desta.

Em 2002 [22], a partir do MA* surgiu o SMA*. O conceito é idêntico, contudo o SMA* utiliza uma estrutura de dados mais eficiente que o MA*.

Algoritmos como o D* [23] ou o D* *lite* [24] têm a capacidade de recalcular a trajetória a cada iteração sem ser necessário refazer todo o processo desde o início. Desta forma, estes algoritmos estão preparados para a hipótese dos custos das ligações ou das posições dos obstáculos serem alterados. O D* *lite* tem uma pequena diferença, uma vez que constrói a trajetória do fim para o início.

2.2.2.2 Estratégias de controlo coordenado

É importante ter em atenção que um ponto no espaço de trabalho pode ser alcançado de três maneiras: movendo apenas o manipulador, movendo apenas a plataforma ou movendo os dois. Na literatura existem vários trabalhos dedicados ao estudo de estratégias para controlo de trajetórias em manipuladores móveis.

Quando uma pessoa pinta um quadro ou uma parede, tende a posicionar o corpo de forma confortável evitando ter o braço completamente esticado, pois seria incómodo e cansativo manter o braço nessa posição. Em vários outros casos, naturalmente o ser humano tende a preferir determinadas posições para o corpo em detrimento de outras. Esta ideia de comportamento pode ser trazida para o estudo dos manipuladores móveis, e foi exatamente isso que foi feito em [25, 26]. A ideia principal por trás destas abordagens é manter o manipulador sempre numa zona de conforto, longe das singularidades. Este conceito de zona de conforto de um manipulador, em termos robóticos é definido pela sua manipulabilidade. Por manipulabilidade entende-se o quão longe uma determinada configuração do manipulador está de atingir uma singularidade, ou seja, o quanto ainda se pode mover sem alcançar configurações singulares, onde o controlo se torna difícil ou mesmo impossível. Assim sendo, o manipulador tenta sempre alcançar a referência como se tivesse em uma base fixa, por sua vez a plataforma móvel tenta acompanhar o manipulador de forma a que ele se mantenha longe das singularidades. Obviamente graças à capacidade de locomoção, existem várias configurações possíveis para posicionar o manipulador móvel. A ideia apresentada

em [26] passa por restringir os movimentos do manipulador em relação ao seu próprio sistema de coordenadas, de forma a que ele se mantenha numa região operacional em que a manipulabilidade é máxima, sendo a plataforma responsável por garantir que o manipulador se mantém dentro desses intervalos.

Segundo [27], considerando um robô com n graus de liberdade, em que as suas juntas são representadas através do vetor q de dimensão n e a posição e orientação da ferramenta de trabalho são definidas por um vetor de dimensão m, com $m \leq n$, a relação entre velocidade da ferramenta de trabalho e a velocidade das juntas pode ser conseguida através da equação seguinte.

$$v = J(q)\dot{q} \tag{2.6}$$

A elipsoide é definida pelo conjunto de todas a velocidades que as juntas conseguem alcançar, de tal forma que a norma de $\dot{q}$,

$$\| \dot{q} \| = \sqrt{(\dot{q}_1^2 + \dot{q}_2^2 + ... + \dot{q}_n^2)} \tag{2.7}$$

satisfaz a seguinte equação.

$$\| \dot{q} \| \leq 1 \tag{2.8}$$

Na direção do maior eixo da elipsoide, o manipulador consegue mover-se com maior velocidade, enquanto na direção do menor eixo o manipulador move-se a velocidades mais baixas. Para calcular a direção dos seus eixos, utiliza-se a decomposição em valores singulares da matriz jacobiana J

$$J = UDV^T \tag{2.9}$$

onde, U e V são matrizes $m \times m$ e $n \times n$ respetivamente e D é uma matriz $m \times n$

$$D = \left[\begin{array}{ccc|c} \sigma_1 & & 0 & \\ & \ddots & & 0 \\ 0 & & \sigma_n & \end{array}\right] \tag{2.10}$$

onde os escalares $\sigma_1, \sigma_2, ..., \sigma_m$ são os valores singulares de J. Sendo u_i a coluna i da matriz U, os eixos da elipsoide de manipulabilidade são dados por $\sigma_1 u_1, \sigma_2 u_2, ..., \sigma_m u_m$. Depois de definida a elipsoide, pode então medir-se a manipulabilidade. Existem dois critérios principais para medir manipulabilidade, que são explicados em [28]. Um desses critérios define a manipulabilidade através de um escalar w dado por

$$w = \sqrt{det(JJ^T)} = \sigma_1 \sigma_2 ... \sigma_m \tag{2.11}$$

onde $\sigma_1 \sigma_2 ... \sigma_m$ corresponde aos valores singulares da matriz jacobiana. Seguindo este critério, w toma um valor proporcional ao volume da elipsoide. Para avaliar a manipulabilidade pode-se avaliar não só o volume da elipsoide, mas também a forma da mesma. Para isso recorre-se à

relação entre o eixo maior e o eixo menor da elipsoide.

$$w = \frac{1}{cond(J)} = \frac{1}{|J|\,|J^{+}|} = \frac{\sigma_{min}}{\sigma_{max}} \tag{2.12}$$

Onde J^{+} corresponde à inversa ou pseudo-inversa de J. Esta técnica, tem a vantagem de normalizar o valor da manipulabilidade, ou seja, $0 \leq w \leq 1$, onde $w = 0$ corresponde a uma configuração singular e $w = 1$ corresponde à melhor configuração para operar.

Nesta abordagem a região operacional do manipulador é limitada pela sua manipulabilidade, contudo podem ser usados outros critérios para escolher essa região. No limite, a opção mais simples é usar o próprio *workspace* como critério, dessa forma se o destino está dentro dos limites do manipulador, então a tarefa é feita movendo apenas o manipulador, caso o destino esteja fora do alcance do manipulador, então a plataforma move-se de forma a trazer o ponto de destino para dentro do *workspace* do manipulador, podendo depois o manipulador mover a ferramenta de trabalho até ao ponto desejado.

Em [29] é proposto um algoritmo de controlo para manipuladores móveis de soldagem. Para isso é proposto um modelo cinemático para os dois subsistemas presentes: a plataforma e o manipulador. Neste trabalho a ferramenta tem de seguir uma trajetória de soldagem com uma velocidade e ângulo constante enquanto a plataforma tem de se mover para evitar as singularidades do manipulador. Eles desenvolveram ainda dois controladores independentes baseados na função de *Lyapunov*.

Similarmente, em [30] é estudada e analisada uma abordagem de controlo independente para coordenar o seguimento de uma trajetória pela ferramenta. Neste trabalho, é dada uma trajetória para a ferramenta de trabalho seguir e com base nessa é definida outra para a plataforma, de tal forma que a trajetória a ser seguida pelo manipulador esteja sempre dentro do seu *workspace*.

Capítulo 3

Cinemática de um manipulador industrial e de uma plataforma omnidirecional

A cinemática de um manipulador é a área da robótica responsável por estudar a velocidade[1] e posição da sua ferramenta de trabalho e das suas juntas. Para se posicionar a ferramenta de trabalho numa determinada posição e orientação é necessário calcular quais coordenadas as juntas devem tomar. O mesmo se passa em relação à velocidade, ou seja, para que a ferramenta se desloque com uma determinada velocidade é necessário calcular qual a velocidade necessária de cada junta.

Esta relação entre a posição da ferramenta e as coordenadas das juntas pode ser estudada de duas formas: através da cinemática direta ou da cinemática inversa. Na cinemática direta calcula-se a posição e orientação da ferramenta de trabalho no espaço cartesiano da base, através das coordenadas das juntas, tratando-se sempre de uma solução possível e única. A cinemática inversa como o próprio nome indica faz o processo inverso, ou seja, sabendo a posição e orientação da ferramenta de trabalho são calculadas as coordenadas de cada uma das juntas, este processo pode ter nenhuma ou várias soluções. Uma vez que o manipulador UR5 é controlado através das coordenadas das suas juntas, o cálculo da cinemática inversa torna-se essencial. Similarmente, a relação entre a velocidade das juntas e a velocidade da ferramenta de trabalho é feita através da matriz jacobiana. O cálculo desta matriz é essencial para se evitar as configurações de singularidade. Os conceitos de cinemática direta e inversa também se aplicam para descrever o movimento dos robôs móveis. Neste caso, a cinemática direta permite perceber qual o movimento do robô tendo em conta a velocidade de cada roda, enquanto a cinemática inversa fornece a velocidade necessária em cada roda, para que o robô se movimente da forma desejada.

[1] Velocidade linear e angular.

3.1 Cinemática direta do manipulador UR5

A cinemática é diretamente influenciada pelo tipo de junta que constituem o manipulador. Uma vez que o UR5 é constituído apenas por juntas rotativas, as variáveis a controlar são as posições angulares de cada junta.

Entre dois referencias diferentes, x e y, pode-se definir a seguinte matriz de transformação:

$$T_x^y = \begin{bmatrix} R_x^y & \vec{P}_x^y \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} x_x & y_x & z_x & (P_x^y)_x \\ x_y & y_y & z_y & (P_x^y)_y \\ x_z & y_z & z_z & (P_x^y)_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

em que R_x^y é a matriz que define a rotação do referencial x para o referencial y e $\vec{P}_x^y$ é o vetor de translação entre estes mesmos referenciais. O objetivo da cinemática direta é calcular a matriz de transformação entre o referencial da base e o da ferramenta, a qual pode ser obtida através da multiplicação sucessiva das matrizes de transformação entre referenciais, ou seja:

$$T_N^0 = T_1^0 \times T_2^1 \times T_3^2 \times \ldots \times T_N^{N-1} \tag{3.2}$$

O método mais comum para o cálculo destas matrizes de transformação é conhecido como método de *Denavit-Hartenberg* [31]. Para aplicar este método primeiramente tem de se atribuir um sistema de coordenadas à base e às juntas do sistema. A atribuição deste sistema de coordenadas deve obedecer às seguintes regras:

1. Z_{i-1} tem de estar ao longo da junta i.

2. X_i deve ser perpendicular e intersetar Z_{i-1}.

3. Y_i completa o sistema.

Para calcular as matrizes de transformação é necessário determinar o valor de 4 parâmetros, conhecidos como parâmetros DH:

- a_i - distância da interseção de Z_{i-1} e X_i à origem do sistema i, medida sobre o eixo X_i.
- α_i - ângulo de rotação em torno de X_i medido de Z_{i-1} a Z_i
- d_i - é a distância desde a origem do sistema i-1 até à interseção dos eixos X_i e Z_{i-1} medida sobre o eixo Z_{i-1}.
- θ_i - ângulo de rotação em torno de Z_{i-1} medido de X_{i-1} a X_i.

A matriz de transformação entre dois referenciais consecutivos fica então definida como:

$$T_i^{i-1} = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i)\times\cos(\alpha_i) & \sin(\theta_i)\times\cos(\alpha_i) & a_i\times\cos(\theta_i) \\ \sin(\theta_i) & \cos(\theta_i)\times\cos(\alpha_i) & -\cos(\theta_i)\times\cos(\alpha_i) & a_i\times\sin(\theta_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

A cinemática do manipulador UR5 tem sido alvo de alguns estudos nos últimos anos, desta forma a solução aqui apresentada é baseada na solução desenvolvida em [32].

Na figura 3.1 pode-se ver a atribuição dos vários referenciais segundo o método DH ao manipulador UR5. Na tabela 3.1 observam-se os valores dos 4 parâmetros DH. Inserindo estes valores

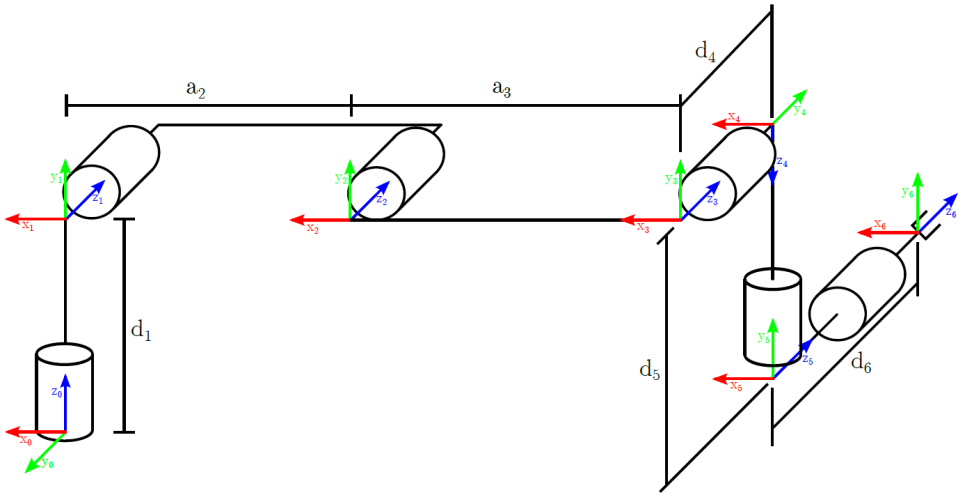


Figura 3.1: Sistema de coordenadas segundo o método DH [32].

Tabela 3.1: Parâmetros de DH para o robô UR5

Junta	a	α	d	θ
1	0	$\pi/2$	0,089159	θ_1
2	-0,425	0	0	θ_2
3	-0,39225	0	0	θ_3
4	0	$\pi/2$	0,10915	θ_4
5	0	$-\pi/2$	0,09465	θ_5
6	0	0	0,0823	θ_6

na equação 3.3, obtém-se as matrizes T_1^0,..., T_6^5.

$$T_1^0 = \begin{bmatrix} R_1^0 & O_1^0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos\theta_1 & 0 & \sin\theta_1 & 0 \\ \sin\theta_1 & 0 & -\cos\theta_1 & 0 \\ 0 & 1 & 0 & 0,08916 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.4a}$$

$$T_2^1 = \begin{bmatrix} R_2^1 & O_2^1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 & 0,425\cos\theta_2 \\ \sin\theta_2 & \cos\theta_2 & 0 & -0,425\sin\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.4b}$$

$$T_3^2 = \begin{bmatrix} R_3^2 & O_3^2 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos\theta_3 & -\sin\theta_3 & 0 & -0,392\cos\theta_3 \\ \sin\theta_3 & \cos\theta_3 & 0 & -0,392\sin\theta_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.4c}$$

$$T_4^3 = \begin{bmatrix} R_4^3 & O_1^0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos\theta_4 & 0 & \sin\theta_4 & 0 \\ \sin\theta_4 & 0 & -\cos\theta_4 & 0 \\ 0 & 1 & 0 & 0,1092 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.4d}$$

$$T_5^4 = \begin{bmatrix} R_5^4 & O_5^4 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos\theta_5 & 0 & -\sin\theta_5 & 0 \\ \sin\theta_5 & 0 & \cos\theta_5 & 0 \\ 0 & -1 & 0 & 0,0947 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.4e}$$

$$T_6^5 = \begin{bmatrix} R_6^5 & O_6^5 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos\theta_6 & -\sin\theta_6 & 0 & 0 \\ \sin\theta_6 & \cos\theta_6 & 0 & 0 \\ 0 & 0 & 1 & 0,0823 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.4f}$$

Finalmente, obtém-se a matriz T_6^0 através das equações 3.2 e 3.4

$$T_6^0 = T_1^0 T_2^1 T_3^2 T_4^3 T_5^4 T_6^5 = \begin{bmatrix} R_6^0 & O_6^0 \\ 0 & 1 \end{bmatrix} \tag{3.5}$$

3.2 Cinemática inversa do manipulador UR5

Como foi referido anteriormente, o objetivo da cinemática inversa passa por descobrir o conjunto de configurações possíveis que satisfazem uma determinada posição e orientação da ferramenta de trabalho.

O primeiro passo é descobrir o valor de θ_1, para isso deve-se considerar a posição da 5ª junta em relação à base, P_5^0. Este ponto pode ser calculado, fazendo uma translação de valor d_6 do referencial 6 para o 5 na direção negativa de eixo z, como se pode ver na figura 3.2. Desta forma tem-se:

$$\vec{P_5^0} = T_6^0 \begin{bmatrix} 0 \\ 0 \\ -d_6 \\ 0 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{3.6}$$

Considerando a figura 3.3 e tendo definido P_5^0, pode-se concluir que:

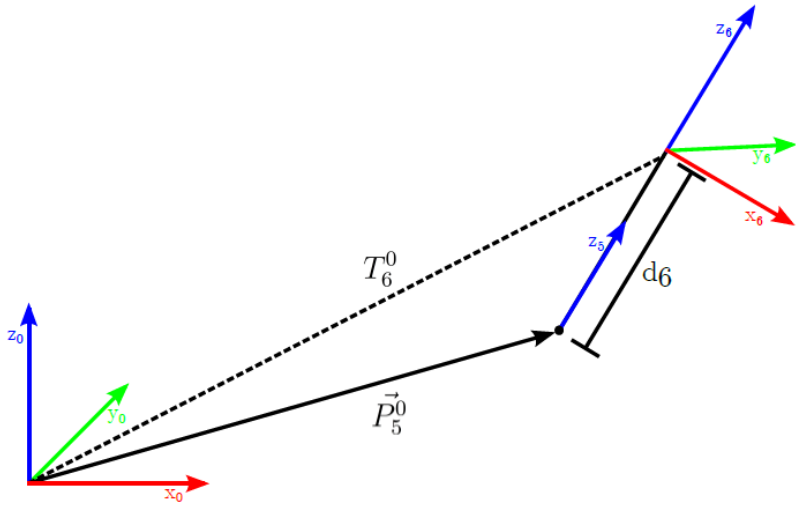


Figura 3.2: Cálculo de P_5^0 [32].

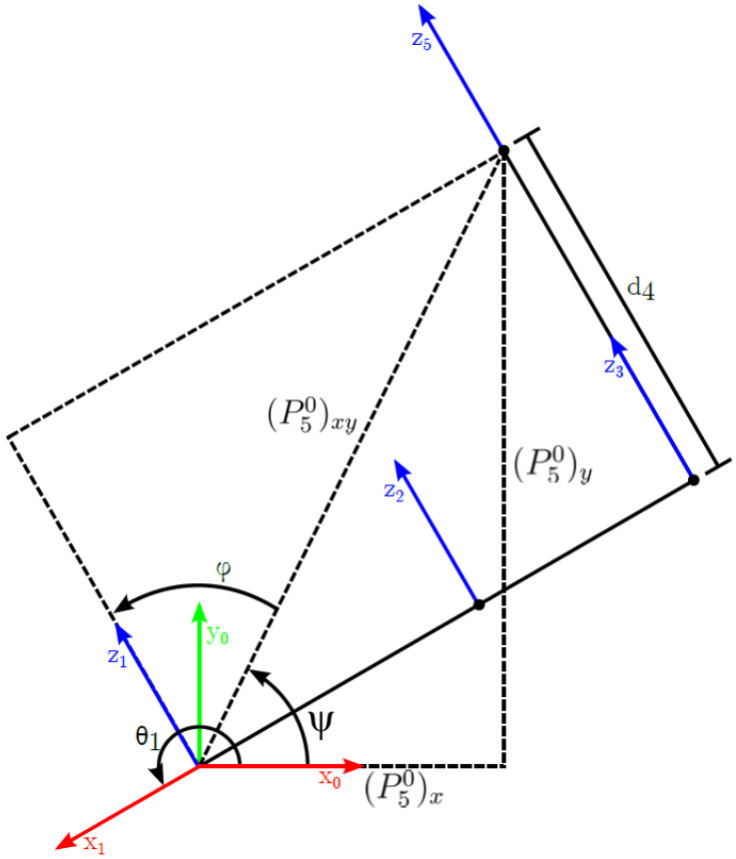


Figura 3.3: Vista superior para o cálculo de θ_1 [32].

$$\theta_1 = \psi + \varphi + \pi/2 \tag{3.7}$$

$$\psi = \text{atan2}((P_5^0)_y, (P_5^0)_x) \tag{3.8}$$

$$\varphi = \pm \arccos\left(\frac{d_4}{\sqrt{(P_5^0)_x{}^2 + (P_5^0)_y{}^2}}\right) \tag{3.9}$$

Depois de calculado o valor de θ_1, recorrendo novamente à vista superior, incluindo agora o referencial 6 como se pode ver na figura 3.4, pode-se calcular θ_5 à custa de P_6^0.

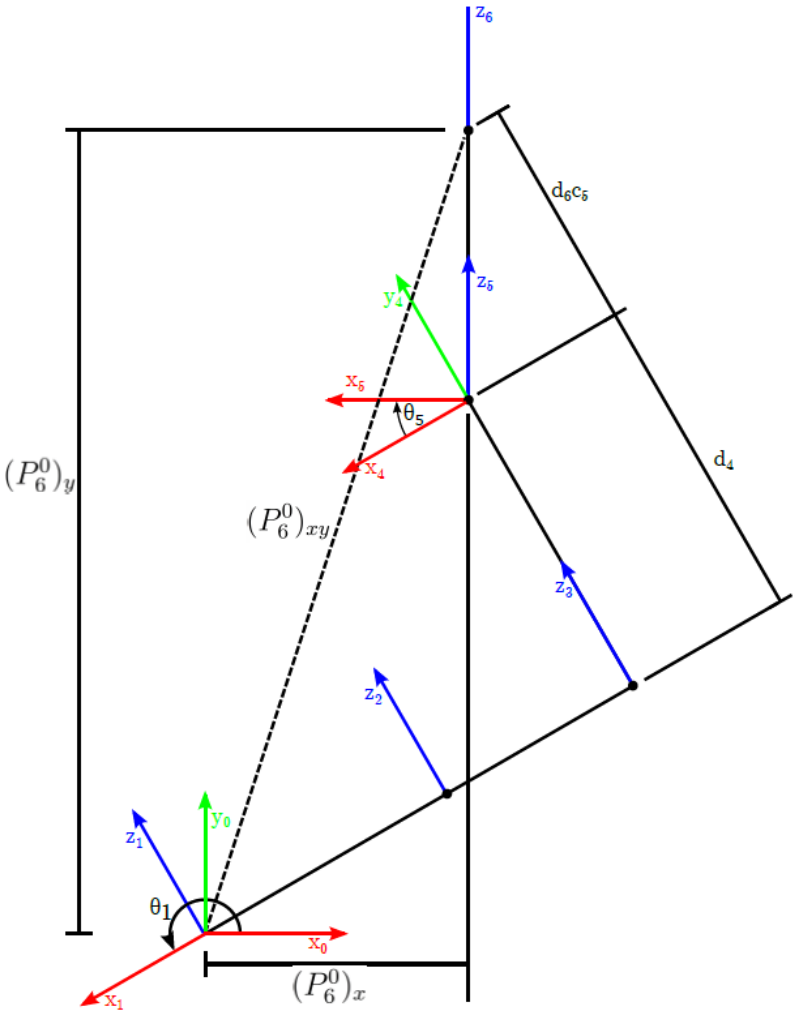


Figura 3.4: Vista superior para o cálculo de θ_5 [32].

Tem-se então:

$$\begin{cases} (P_6^1)_z &= d_6 \cos\theta_5 + d_4 \\ (P_6^1)_z &= (P_6^0)_x \sin\theta_1 - (P_6^0)_y \cos\theta_1 \end{cases} \tag{3.10}$$

Das equações (3.10) retira-se:

$$\theta_5 = \pm \arccos\frac{(P_6^1)_z - d_4}{d_6} \tag{3.11}$$

Como se pode ver na figura 3.5 as componentes x e y do vetor z_1, respetivamente z_x e z_y, estão

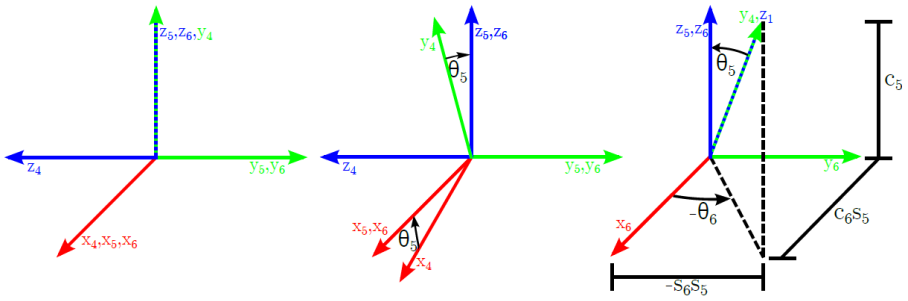


Figura 3.5: Cálculo de θ_6 [32].

relacionadas com o plano xy do referencial 6 através de θ_5 e θ_6. Para se calcular z_x e z_y, é necessário calcular a matriz de transformação homogénea entre o referencial 6 e 1.

$$T_1^6 = ((T_1^0)^{-1}.T_6^0)^{-1} \tag{3.12}$$

Recorrendo à matriz (3.1) e à figura 3.5 obtém-se:

$$\begin{cases} z_x = \sin\theta_5 \cos\theta_6 \\ z_y = -\sin\theta_6 \sin\theta_5 \end{cases} \tag{3.13}$$

De (3.13) retira-se:

$$\theta_6 = \text{atan2}\left(\frac{-z_y}{\sin\theta_5}, \frac{z_x}{\sin\theta_5}\right) \tag{3.14}$$

As restantes juntas podem ser calculadas considerando que estas formam um manipulador planar com 3 juntas rotativas. Deve-se primeiramente calcular $\vec{P_3^1}$.

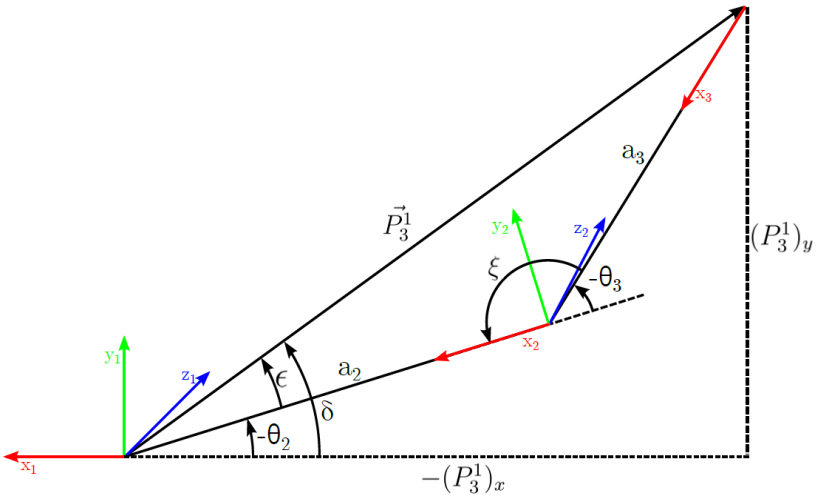


Figura 3.6: Plano que contém referencial 1, 2 e 3 [32].

$$T_4^1 = T_6^1 T_4^6 = T_6^1 \left(T_5^4 T_6^5\right)^{-1} \tag{3.15}$$

$$\vec{P_3^1} = T_4^1 \begin{bmatrix} 0 \\ -d_4 \\ 0 \\ 1 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{3.16}$$

Analisando a figura 3.6 denota-se que há uma relação entre $\vec{P_3^1}$, θ_3, a_2 e a_1:

$$\cos\xi = -\cos(\pi - \xi) = -\cos(-\theta_3) = \cos\theta_3 \tag{3.17}$$

$$\begin{cases} (P_3^1)_y = a_2\sin(-\theta_2) + a_3\sin(-\theta_2 - \theta_3) \\ -(P_3^1)_x = a_2\cos(-\theta_2) + a_3\cos(-\theta_2 - \theta_3) \end{cases} \tag{3.18}$$

Relacionando (3.17) e (3.18):

$$\theta_3 = \pm\arccos\left(\frac{(P_3^1)_x{}^2 + (P_3^1)_y{}^2 - a_2^2 - a_3^2}{2a_2a_3}\right) \tag{3.19}$$

A figura 3.6 também mostra que:

$$\theta_2 = -(\delta - \varepsilon) \tag{3.20}$$

$$\delta = \text{atan2}\left((P_3^1)_y, -(P_3^1)_x\right) \tag{3.21}$$

$$\varepsilon = \text{atan2}\left(a_2 + a_3\cos\theta_3, a_3\sin\theta_3\right) \tag{3.22}$$

Relacionando as equações (3.21) e (3.22):

$$\theta_2 = -\text{atan2}\left((P_3^1)_y, -(P_3^1)_x\right) + \text{atan2}\left(a_2 + a_3\cos\theta_3, a_3\sin\theta_3\right) \tag{3.23}$$

Falta apenas calcular θ_4, para tal é necessário calcular T_4^3:

$$T_4^3 = T_1^3\, T_4^1 = \left(T_2^1\, T_3^2\right)^{-1} T_4^1 \tag{3.24}$$

Observando as matrizes (3.3) e (3.1) obtém-se:

$$\theta_4 = \text{atan2}\left(x_y, x_x\right) \tag{3.25}$$

Pela análise dos resultados, percebe-se que a cinemática inversa tem 2^3 soluções possíveis. Isso deve-se às várias configurações que o manipulador pode tomar, que resultam das combinações ombro à esquerda, ombro à direta, cotovelo para cima, cotovelo para baixo, punho para cima e punho para baixo (figura 3.7).

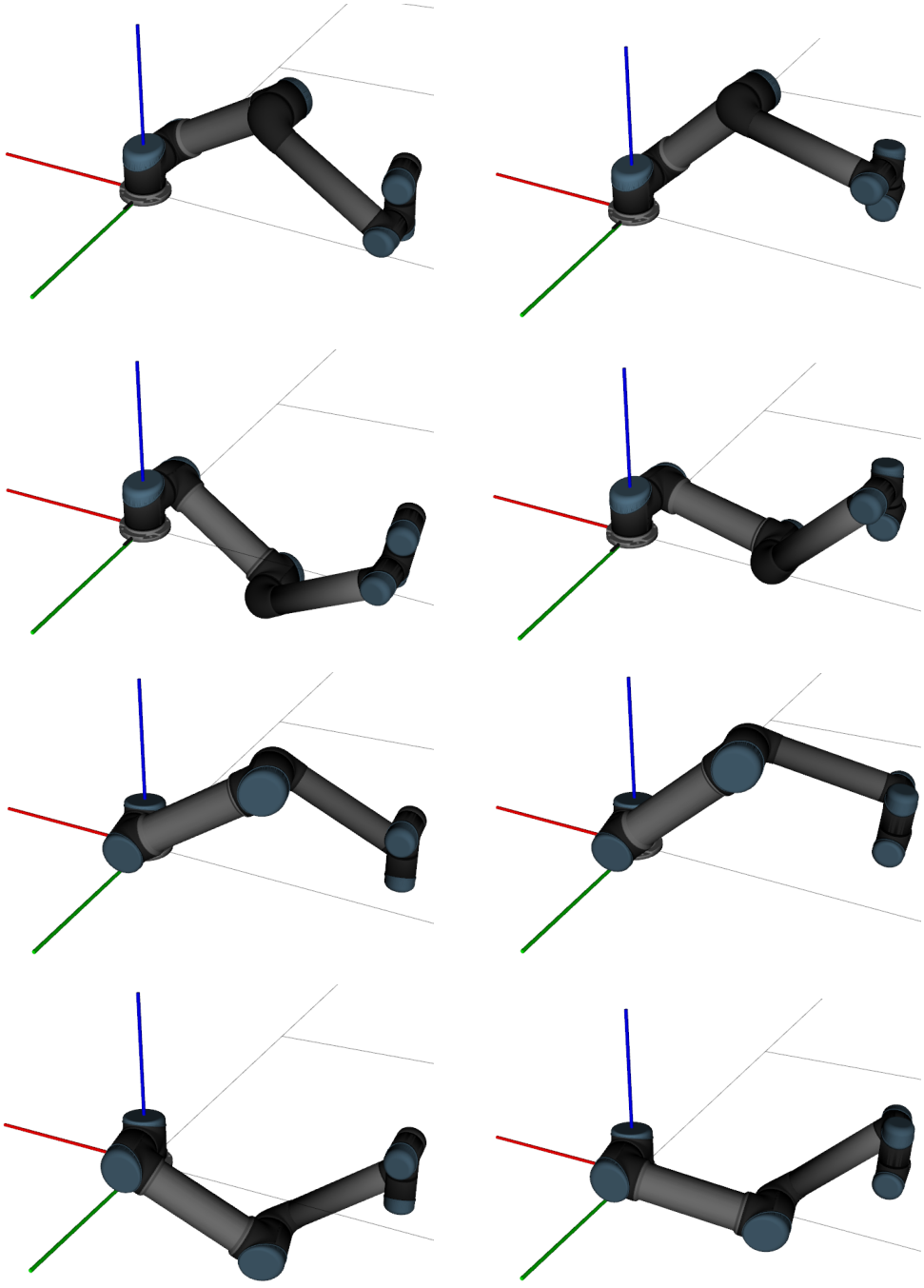


Figura 3.7: Conjunto de configurações do manipulador UR5.

3.3 Jacobiano

Pretende-se agora determinar a relação entre a velocidade linear e angular, v_{ef} e w_{ef} respetivamente, da ferramenta em relação ao referencial da base e expressas no referencial da base, com as velocidades linear e angular das juntas, $\dot{q}$,

$$\begin{bmatrix} v_{ef} \\ w_{ef} \end{bmatrix} = \begin{bmatrix} J_v \\ J_W \end{bmatrix} \dot{q} \tag{3.26}$$

em que J_v e J_w são matrizes 3x6 que representam a contribuição da velocidade de rotação das juntas para a velocidade linear a angular da ferramenta de trabalho, respetivamente. A parte angular do jacobiano, J_w, pode ser calculada através de

$$J_w = \begin{bmatrix} z_0^0 & z_1^0 & z_2^0 & z_3^0 & z_4^0 & z_5^0 \end{bmatrix} \tag{3.27}$$

onde

$$\begin{aligned} z_0^0 &= k \\ z_1^0 &= R_1^0 k \\ z_2^0 &= R_2^0 k = R_1^0 R_2^1 k \\ z_3^0 &= R_2^0 k = R_1^0 R_2^1 R_3^2 k \\ z_4^0 &= R_2^0 k = R_1^0 R_2^1 R_3^2 R_4^3 k \\ z_5^0 &= R_2^0 k = R_1^0 R_2^1 R_3^2 R_4^3 R_5^4 k \end{aligned} \tag{3.28}$$

com $k = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$ e onde R_{i+1}^i $i \in 1, \ldots, 4$ é dado pela equação 3.4.
Por sua vez, a parte linear do jacobiano, J_v, é dada por

$$J_{vi} = z_{i-1}^0 \left(O_6^0 - O_{i-1}^0 \right) = \frac{\partial O_6^0}{\partial q_i} \tag{3.29}$$

onde O_6^0 é o vetor da 4ª coluna da matriz T_6^0 obtida através da equação 3.5. O jacobiano fica então dado por

$$J = \begin{bmatrix} J_v \\ J_w \end{bmatrix} = \begin{bmatrix} \frac{\partial O_6^0}{\partial q_1} & \frac{\partial O_6^0}{\partial q_2} & \frac{\partial O_6^0}{\partial q_3} & \frac{\partial O_6^0}{\partial q_4} & \frac{\partial O_6^0}{\partial q_5} & \frac{\partial O_6^0}{\partial q_6} \\ z_0^0 & z_1^0 & z_2^0 & z_3^0 & z_4^0 & z_5^0 \end{bmatrix} \tag{3.30}$$

3.3.1 Manipulabilidade

Para este projeto a manipulabilidade foi definida como um objetivo de otimização, de forma a maximizar a *perfomance* do manipulador. Baseado no jacobiano pode-se definir a manipulabilidade de acordo com as equações (2.11) e (2.12). Contudo, uma vez que UR5 possui 6 juntas, o jacobiano resulta em uma matriz 6×6. Combinando as equações (2.6) e (2.8) obtém-se

$$v^T \left(J(q) J_q^T \right)^{-1} v = 1 \tag{3.31}$$

onde v representa a velocidade espacial da ferramenta de trabalho, sendo os 3 primeiras elementos referentes à velocidade de translação e os restantes 3 à velocidade de rotação. A equação (3.31) representa os pontos de uma elipse de 6 dimensões, assim sendo, a manipulabilidade do UR5 pode ser dividida em manipulabilidade de translação e manipulabilidade de rotação, resultando assim duas elipses, possíveis de representar graficamente [33].

3.4 Cinemática do veículo omnidirecional

Na figura 3.8 pode ver-se uma representação de um robô omnidirecional com 4 rodas *mecanum* que ajudará no desenvolvimento das equações da cinemática.

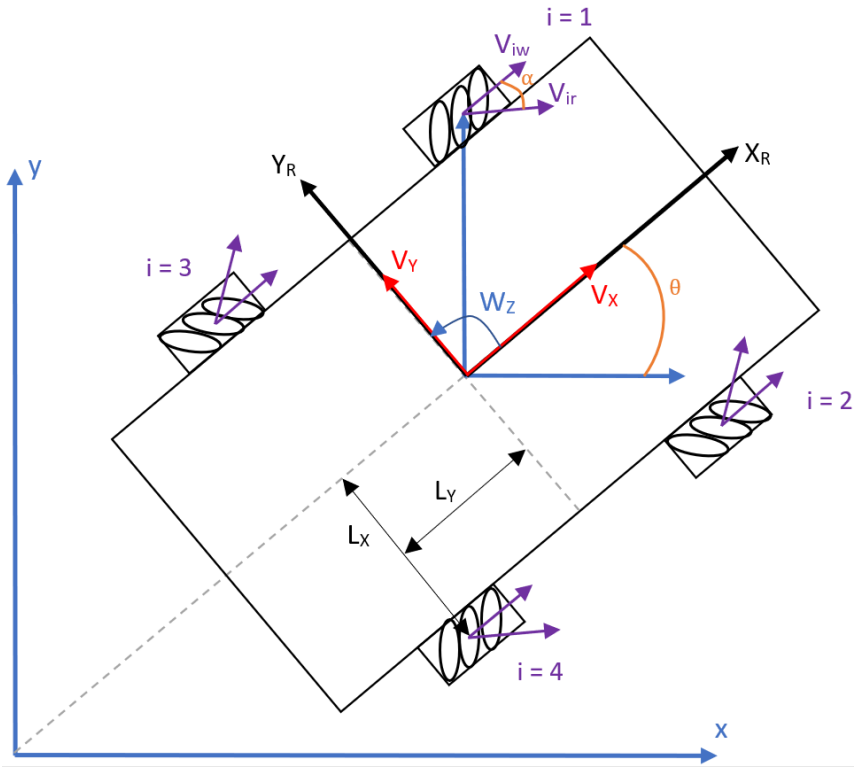


Figura 3.8: Configuração omnidirecional com 4 rodas *mecanum*.

- x,y corresponde ao referencial global.
- X_R, Y_R corresponde ao referencial do robô.
- $V_X, V_Y\ [m/s] \in R$ corresponde à velocidade linear do robô.
- $W_Z\ [rad/s] \in R$ corresponde à velocidade angular do robô.
- $V_{iw}\ [m/s]\ (i = 1,2,3,4) \in R$ é o vetor de velocidade correspondente à rotação das rodas, onde $V_{iw} = R_w * W_{iw}$.
- R_w corresponde ao raio das rodas e $W_{iw}[rad/s]$ à velocidade angular da roda.

- $V_{ir}\,[m/s]\,(i = 1,2,3,4) \in R$ é o vetor de velocidade perpendicular ao eixo de rotação do rolo em contacto com o piso.

- $V_{iX}, V_{iY}\,[m/s]\,(i = 1,2,3,4) \in R$ são as velocidades resultantes na roda, em relação ao referencial do robô.

- α é o ângulo entre o eixo da roda e o eixo do rolo.

- L_x, L_y corresponde às distâncias entre o centro do robô e o centro da roda, no eixo X_R e Y_R respetivamente.

Considerando $\alpha = \frac{\pi}{4}$, pode-se definir a velocidade resultante em cada roda como:

$$\begin{aligned}
V_{1x} &= V_{1w} + V_{1r}\sin\frac{\pi}{4}\ ,\ V_{1y} = V_{1r}\sin\frac{\pi}{4}\\
V_{2x} &= V_{2w} + V_{2r}\sin\frac{\pi}{4}\ ,\ V_{2y} = V_{2r}\sin\frac{\pi}{4}\\
V_{3x} &= V_{3w} + V_{3r}\sin\frac{\pi}{4}\ ,\ V_{3y} = V_{3r}\sin\frac{\pi}{4}\\
V_{4x} &= V_{4w} + V_{4r}\sin\frac{\pi}{4}\ ,\ V_{4y} = V_{4r}\sin\frac{\pi}{4}
\end{aligned} \tag{3.32}$$

Uma vez que as rodas estão presas à plataforma, a relação entre a velocidade das rodas e a velocidade da plataforma é a seguinte:

$$\begin{aligned}
V_{1x} &= V_X - L_x * W_z\ ,\ V_{1y} = V_Y + L_Y * W_Z\\
V_{2x} &= V_X + L_x * W_z\ ,\ V_{2y} = V_Y + L_Y * W_Z\\
V_{3x} &= V_X - L_x * W_z\ ,\ V_{3y} = V_Y - L_Y * W_Z\\
V_{4x} &= V_X + L_x * W_z\ ,\ V_{4y} = V_Y - L_Y * W_Z
\end{aligned} \tag{3.33}$$

Relacionando as equações 3.32 e 3.33 chega-se à equação da cinemática inversa da plataforma.

$$\begin{bmatrix} V_{1w}\\ V_{2w}\\ V_{3w}\\ V_{4w} \end{bmatrix} = \begin{bmatrix} 1 & -1 & -(L_x+L_y)\\ 1 & 1 & (L_x+L_y)\\ 1 & 1 & -(L_x+L_y)\\ 1 & -1 & (L_x+L_y) \end{bmatrix} \begin{bmatrix} V_X\\ V_Y\\ W_Z \end{bmatrix} \tag{3.34}$$

Fazendo um tratamento algébrico à equação (3.34) e recorrendo-se à pseudo-inversa obtém-se a cinemática direta, dada pela equação 3.35.

$$\begin{bmatrix} V_X\\ V_Y\\ W_Z \end{bmatrix} = \frac{1}{4} \begin{bmatrix} 1 & 1 & 1 & 1\\ -1 & 1 & 1 & -1\\ -\frac{1}{L_x+L_y} & \frac{1}{L_x+L_y} & -\frac{1}{L_x+L_y} & \frac{1}{L_x+L_y} \end{bmatrix} \begin{bmatrix} V_{1w}\\ V_{2w}\\ V_{3w}\\ V_{4w} \end{bmatrix} \tag{3.35}$$

Finalmente, a velocidade no referencial global pode ser obtida através da equação 3.36.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} V_X \\ V_Y \\ W_z \end{bmatrix} \tag{3.36}$$

Capítulo 4

Cinemática composta de um manipulador móvel

Para o controlo do manipulador móvel são propostas duas abordagens. Na primeira é feito um controlo baseado em dois subsistemas separados, onde a plataforma móvel e o manipulador são controlados de forma independente um do outro, através da cinemática desenvolvida no capítulo 3. A segunda abordagem trata o sistema como um todo, sendo este resultante da integração do manipulador com a plataforma móvel. Este sistema composto tem como resultado um manipulador de 9 juntas, cuja cinemática é resolvida com recurso ao `trac_ik`, sendo esta uma biblioteca *ROS* que será explicada em mais detalhe na secção 4.5.1.

Para o desenvolvimento desta dissertação é ainda necessário recorrer a algumas ferramentas que ajudam o desenvolvimento e a implementação de soluções para o problema em questão. Essas ferramentas serão abordadas no início do presente capítulo. Serão ainda explicados alguns pontos sobre o controlo da plataforma e do manipulador, sendo posteriormente abordadas as lógicas implementadas. Por fim será feita uma comparação dos resultados obtidos.

4.1 Ferramentas

Para otimização de tempo e custo torna-se imprescindível recorrer a um simulador que possa reproduzir o comportamento do manipulador móvel em questão. Para tal, é usado o simulador *V-REP*.

Para se desenvolver uma arquitetura modular, que sustente o desenvolvimento de um projeto complexo é usada a *framework ROS*. Nesta secção é ainda explicada a relação entre o *V-REP* e o *ROS*, tendo em vista o controlo do cenário do *V-REP* através de uma aplicação externa.

4.1.1 *Virtual Robot Experimentation Platform* (V-REP)

O *V-REP* trata-se de um simulador que permite a simulação de robôs em ambientes terrestres, aéreos ou aquáticos. Além disso, permite dividir um sistema complexo em um conjunto de objetos mais simples que podem ser programados de forma independente uns dos outros. Diversos

modelos de robôs já estão disponíveis no simulador para utilização, contudo o utilizador pode desenvolver os seus próprios robôs e cenários.

O manipulador UR5 e a plataforma omnidirecional já se encontram disponíveis no *V-REP*, não sendo necessário o seu desenvolvimento. A plataforma omnidirecional com 4 rodas *mecanum* disponível no *V-REP* trata-se de uma plataforma desenvolvida pela *KUKA Industrial Robots* que se ajusta perfeitamente ao caso em estudo. Na figura 4.1 pode ver-se o manipulador móvel utilizado

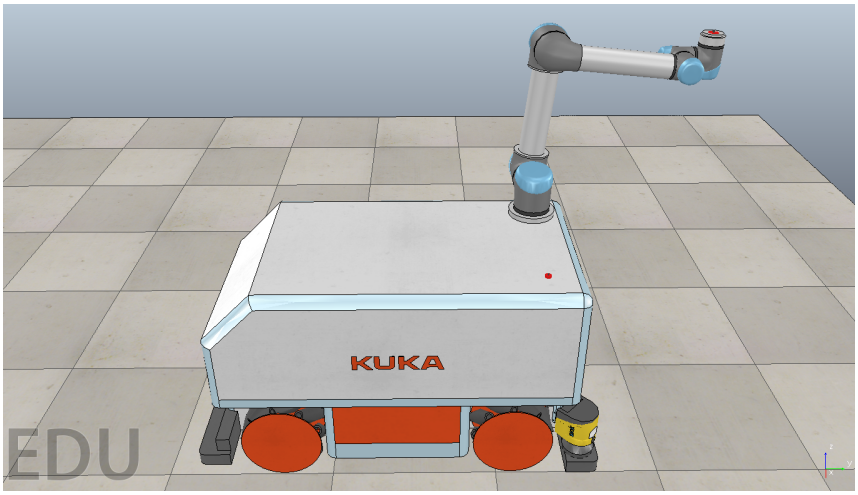


Figura 4.1: Manipulador móvel.

nesta dissertação.

A simulação do *V-REP* pode ser controlada através de *scripts* internos que são programados em *LUA*. Contudo, uma vez que se pretende desenvolver um sistema modular, é utilizada a interface *ROS* disponibilizada pelo mesmo, que faz o *V-REP* funcionar como um nó do *ROS*, permitindo a sua comunicação com outros nós. Desta forma todo o controlo é desenvolvido em C++ numa aplicação externa ao *V-REP*, sendo o *ROS* responsável por efetuar a ligação entre a aplicação e este.

4.1.2 *Robot Operation System* (ROS)

O *ROS* trata-se de uma *framework* que fornece bibliotecas e ferramentas para desenvolvimento de aplicações em robótica e que foi desenvolvida para ser o mais distribuída e modular possível, permitindo a reutilização de código e recursos.

O funcionamento do *ROS* é baseado num sistema de nós, independentes uns dos outros e que comunicam entre si através de mensagens (usando o modelo *publisher/subscriber*), como se fossem programas a executar paralelamente. Um nó publica uma mensagem num tópico e qualquer nó pode subscrever esse tópico para ter acesso à mensagem. Uma das vantagens deste sistema de nós é o facto de que os nós do *ROS* não precisam de estar na mesma máquina. Desta forma, desenvolver uma solução em *ROS* permite dividir um problema complexo em vários nós, sendo cada um destes responsável por solucionar uma parte mais simples do problema geral. Cada nó pode escrever e subscrever vários tópicos, assim como, cada tópico pode ter vários nós a publicar

e a subscrever nele. Na figura 4.2 está representada a comunicação entre vários nós através de um tópico. Caso um nó precise de receber uma resposta a uma mensagem sua, deve usar-se serviços

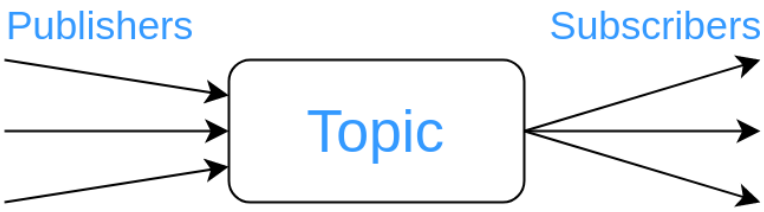


Figura 4.2: Funcionamento de um tópico.

em vez de tópicos.

Um serviço é algo disponibilizado por um nó e que pode ser requisitado diretamente por outro nó, para isso é necessário a troca de duas mensagens: uma para enviar o pedido e outra para receber a resposta. Um nó disponibiliza o serviço através de um determinado nome (definido por uma *string*), o cliente chama esse serviço através desse nome enviando uma mensagem de pedido e fica a aguardar a resposta. Este processo está exemplificado na figura 4.3. Como explicado

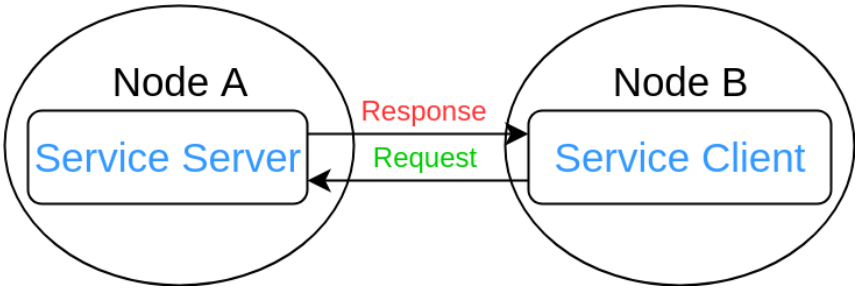


Figura 4.3: Funcionamento de um serviço.

no tópico anterior, a comunicação entre o programa de controlo e o *V-REP* é efetuada através do *ROS*. Para isso, o *V-REP* funciona como um nó do *ROS* que subscreve os tópicos onde a informação relativa aos objetos existentes no *V-REP* é publicada, como é o caso da velocidade das rodas da plataforma móvel ou as coordenadas das juntas do manipulador. Depois de recebida essa informação através do tópico, as funções de *callback* de cada tópico tratam de a processar, fazendo as respetivas alterações no cenário. O *V-REP* também publica em tópicos informação necessária acerca da simulação, como por exemplo, a posição atual do robô ou as coordenadas atuais das juntas do manipulador. Na figura 4.4 pode ver-se a estrutura que é usada para comunicar com o *V-REP*. O programa de controlo inicia um nó chamado "/MainControl" que em cada ciclo de controlo publica nos tópicos "/moveUr5To" e "/moveBaseTo" o valor das juntas desejadas para o manipulador e a posição desejada para a plataforma, respetivamente. O nó "/vrep_ros_interface" subscreve esses tópicos e está constantemente a publicar o que recebe nos tópicos inicializados no *V-REP*. Seguidamente recorrendo aos *scripts* internos do *V-REP*, os valores publicados nestes tópicos são lidos e enviados para o manipulador e para a plataforma móvel. Esta figura foi obtida

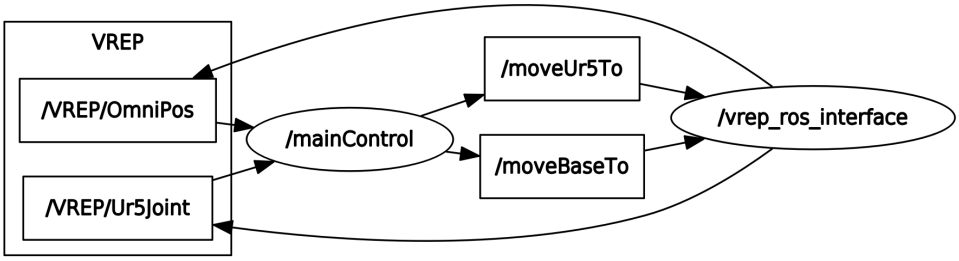


Figura 4.4: Comunicação com V-REP via ROS.

através do `rqt_graph`, que se trata de uma ferramenta disponibilizada pelo *ROS* para visualizar todos os nós e tópicos que estão a executar no *ROS* e as relações entre eles.

4.2 Algoritmo de navegação da plataforma móvel

Para efetuar a navegação da plataforma pelo mundo é usado o sistema de troca de mensagens explicado anteriormente, onde o programa de controlo em cada ciclo indica ao manipulador a velocidade desejada para as 4 rodas da plataforma. Por sua vez, o simulador em cada ciclo publica a posição e orientação da plataforma no mundo, como se pode ver na figura 4.4.

Uma vez que a capacidade omnidirecional da plataforma móvel lhe permite mover-se em qualquer direção, sabendo a posição[1] atual da mesma e a posição desejada, é possível calcular as velocidades das rodas que fazem a plataforma mover-se na direção pretendida (figura 4.5).

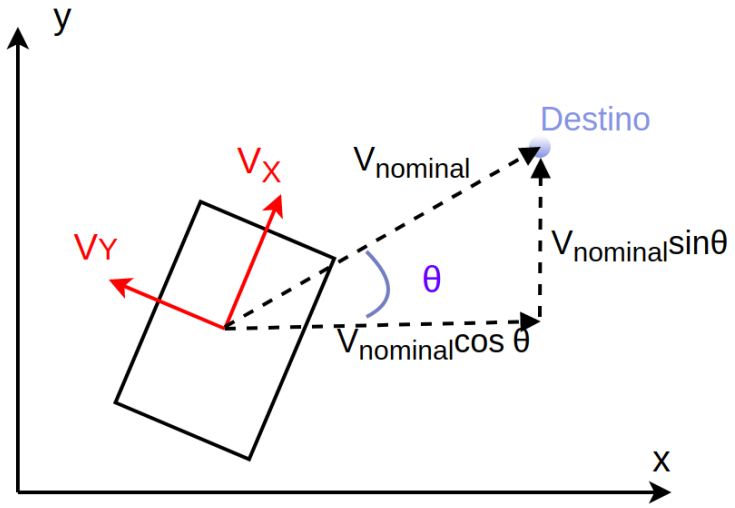


Figura 4.5: Cálculo da velocidade em x e y.

Tendo a velocidade $\dot{x}$ e $\dot{y}$, através das equações (3.36) e (3.34) obtém-se os valores das velocidades das rodas necessários para que o robô se mova em direção ao destino.

O algoritmo 1 calcula a velocidade das rodas v_1, v_2, v_3, v_4 necessárias para levar o robô desde a posição atual $(X_{atual}, Y_{atual}, angle_{atual})$ até à posição desejada $(X_{dest}, Y_{dest}, angle_{dest})$. Considerando que a plataforma se move à velocidade nominal V_{nom}, calculando o ângulo entre o ponto de destino e o centro da plataforma, é possível calcular as velocidades V_x e V_y que levam a plataforma até ao ponto de destino. As variáveis $erro_{dist}$ e $erro_{angle}$ representam, respetivamente, o erro de distância e orientação entre o ponto atual da plataforma e o ponto de destino. Se esses erros estiverem dentro de um *threshold* então a plataforma está no ponto de destino, caso contrário é preciso calcular as velocidades das rodas recorrendo à função $OmniIk()$ que implementa as equações explicadas na secção 3.4. A função $NormalizeAngle()$ é responsável por fazer a normalização do ângulo que lhe for passado como parâmetro. Já a função $testDir()$ retorna 1 ou -1 e é responsável por fazer a plataforma rodar no sentido que minimiza o movimento do robô.

[1] Posição é definida por (x, y, θ)

Algoritmo 1: Função que calcula as velocidades necessárias das rodas para mover a plataforma até uma posição com determinada orientação.

Input : $X_{dest}, Y_{dest}, angle_{dest}, X_{atual}, Y_{atual}, angle_{atual}$
Output: v_1, v_2, v_3, v_4

begin
  $erro_{dist} = \sqrt{(X_{dest} - X_{atual})^2 + (Y_{dest} - Y_{atual})^2}$;
  $erro_{angle} = abs(NormalizeAngle(angle_{dest} - angle_{atual}))$;
  $angle = \text{atan2}(y_{dest} - y_{atual}, x_{dest} - x_{atual})$;
  $V_x = V_{nom} \times \cos(angle)$;
  $V_y = V_{nom} \times \sin(angle)$;
  if $erro_{dist} > threshold_{dist}$ *ou* $erro_{angle} > threshold_{angle}$ **then**
    if $erro_{dist} > threshold_{dist}$ **then**
      $[v_1, v_2, v_3, v_4] = OmniIk(V_x, V_y, 0, omniOri)$;
    end
    if $erro_{angle} > threshold_{angle}$ **then**
      $[v_1, v_2, v_3, v_4] = [v_1, v_2, v_3, v_4] + OmniIk(0, 0, testDir() \times V_{rot}, omniOri)$;
    end
  else
    $v_1 = 0$;
    $v_2 = 0$;
    $v_3 = 0$;
    $v_4 = 0$;
  end
end

Para se testar o algoritmo desenvolvido, colocou-se a plataforma móvel na posição inicial (0,0,0) e definiu-se a posição de destino (1.5, 2, $\frac{\pi}{2}$). Nas figuras 4.6 e 4.7 pode ver-se a navegação do robô até atingir a referência.

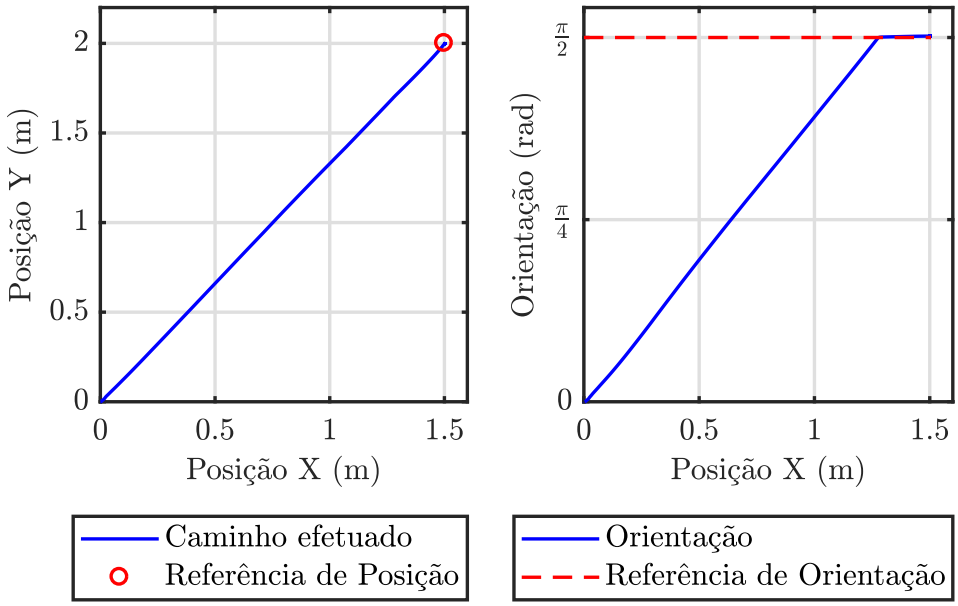


Figura 4.6: Evolução da posição e orientação da plataforma.

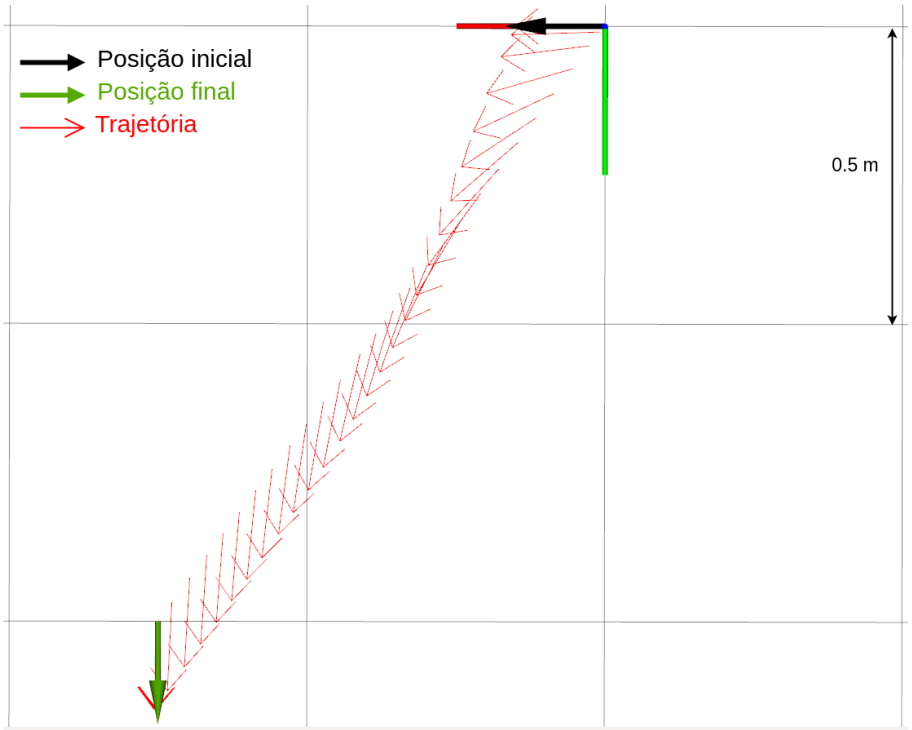


Figura 4.7: Trajetória efetuada pela plataforma.

4.3 Controlo do manipulador

A cinemática inversa do manipulador é feita em relação ao referencial do mesmo. Contudo, uma vez que este está em movimento o seu referencial também está, desta forma, para o manipulador alcançar um ponto no mapa, definido pelas suas coordenadas no mundo, é necessário efetuar a conversão dessas coordenadas para coordenadas no referencial do manipulador (figura 4.8).

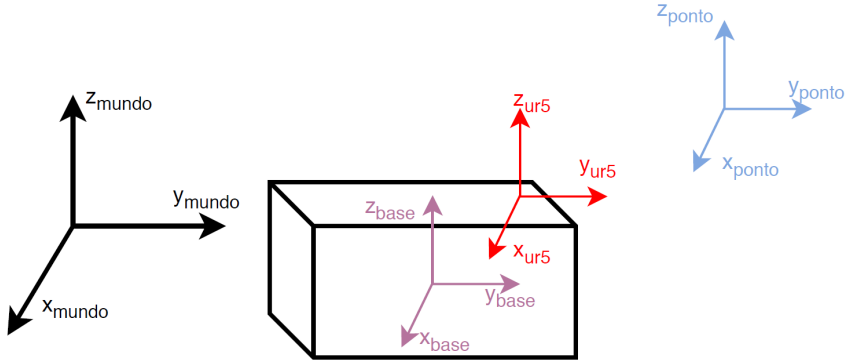


Figura 4.8: Relações entre os referenciais do sistema.

Recorrendo à equação (3.2), pode definir-se a matriz homogénea do ponto de destino, T_{ponto}^{mundo}, como

$$T_{ponto}^{mundo} = T_{base}^{mundo} \times T_{ur5}^{base} \times T_{ponto}^{ur5} \tag{4.1}$$

Através de algumas transformações algébricas, tem-se a seguinte equação

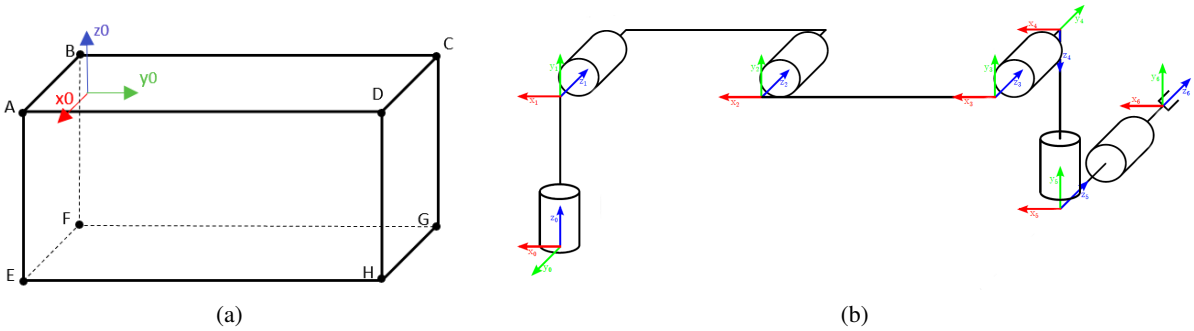
$$T_{ponto}^{ur5} = (T_{ur5}^{base})^{-1} \times (T_{base}^{mundo})^{-1} \times T_{ponto}^{mundo} \tag{4.2}$$

em que a matriz T_{ur5}^{base} é constante e apenas depende da montagem física do manipulador na plataforma e T_{base}^{mundo} é calculada através dos dados da posição e orientação da plataforma fornecidos pelo simulador através do *ROS*.

Sabendo T_{ur5}^{base}, pode aplicar-se as equações da secção 3.2 para obter as coordenadas das juntas que fazem o manipulador alcançar o ponto de destino com a orientação desejada. Contudo, estas equações não têm em conta o facto do manipulador estar anexado a uma plataforma, o que faz com que exista uma área que não pode ser invadida. É então necessário garantir que nenhum dos elos (que formam o manipulador) invada esta área, ou seja, é necessário garantir que o manipulador não colida com a plataforma omnidirecional.

Através das equações (3.4) e (3.2) é possível calcular as coordenadas de cada junta em relação ao referencial do manipulador. Definindo os segmentos de reta que unem as juntas e sabendo a posição do manipulador em relação à plataforma, consegue-se perceber se algum desses segmentos invade algum dos planos que definem o espaço ocupado pela plataforma. Caso isso aconteça, então essa solução da cinemática inversa deve ser descartada.

Tomando como exemplo um dos planos que delimitam o espaço da plataforma, observando a figura 4.9, considera-se as coordenadas do ponto A, B, C e D em relação ao referencial $x_0y_0z_0$



(a) (b)

Figura 4.9: Plataforma e manipulador.

como sendo $(A_x, A_y, 0)$, $(B_x, B_y, 0)$, $(C_x, C_y, 0)$ e $(D_x, D_y, 0)$, respetivamente. Estes pontos definem um semiplano em $z = 0$ delimitado pelos segmentos de reta $\overline{AB}$, $\overline{BC}$, $\overline{CD}$ e $\overline{DA}$, como se pode comprovar na figura 4.9a. O raciocínio aplicado para este plano replica-se para os restantes.

Um segmento de reta é definido pela seguinte equação vetorial,

$$P(t) \;=\; (1-t)\,P_1 + tP_2 \;,\; 0 \leq t \leq 1 \tag{4.3}$$

em que P_1 e P_2 são os pontos inicial e final do segmento de reta. Para definir um plano recorre-se à equação (4.4).

$$ax + by + cz + d = 0 \tag{4.4}$$

Considerando o segmento de reta definido por dois pontos P_1 e P_2 como (P_{1x}, P_{1y}, P_{1z}) e (P_{2x}, P_{2y}, P_{2z}) respetivamente, desenvolvendo a equação (4.3) e considerando $V = P_2 - P_1$ obtém-se a seguinte equação.

$$P(t) \;=\; (P_{1x} + tV_x, P_{1y} + tV_y, P_{1z} + tV_z) \tag{4.5}$$

Relacionando as equações (4.4) e (4.5) resulta

$$\begin{aligned}
&a(P_{1x} + tV_x) + b(P_{1y} + tV_y) + c(P_{1z} + tV_z) + d = 0 \\
&\begin{bmatrix} a & d & c \end{bmatrix} \begin{bmatrix} P_{1x} & V_x \\ P_{1y} & V_y \\ P_{1z} & V_z \end{bmatrix} \begin{bmatrix} 1 \\ t \end{bmatrix} = -d \\
&k \begin{bmatrix} 1 \\ t \end{bmatrix} = -d \\
&\begin{bmatrix} 1 \\ t \end{bmatrix} = -k^{-1} d
\end{aligned} \tag{4.6}$$

Calculado t, caso não verifique a inequação $0 \leq t \leq 1$, então o segmento de reta não interseta o plano, caso contrário, substituindo t na equação (4.3) obtém-se o ponto de interseção do segmento

com o plano. Caso esse ponto esteja dentro dos limites definidos pelos segmentos de reta que delimitam o plano, então o segmento de reta interseta-o, ou seja, essa solução da cinemática inversa não é válida.

Algoritmo 2: Função que verifica se o segmento de reta definido por dois pontos interseta um plano.

Input : $P_1, P_2, B, Xmax, Xmin, Ymax, Ymin, Zmax, Zmin$
Output: result

```
1  begin
2      Definir equação da reta;
3      Definir equação do plano;
4      Definir a matriz k através das coordenadas do ponto P1, do plano B e do vetor P2 − P1;
5      Calcular t;
6      if t > 1 ou t < 0 then
7          result = false;
8          return result;
9      else
10         Calcular ponto de interseção P_intersecao;
11         if P_intersecao está dentro dos limites do plano then
12             result = true;
13             return result;
14         else
15             result = false;
16             return result;
17         end
18     end
19 end
```

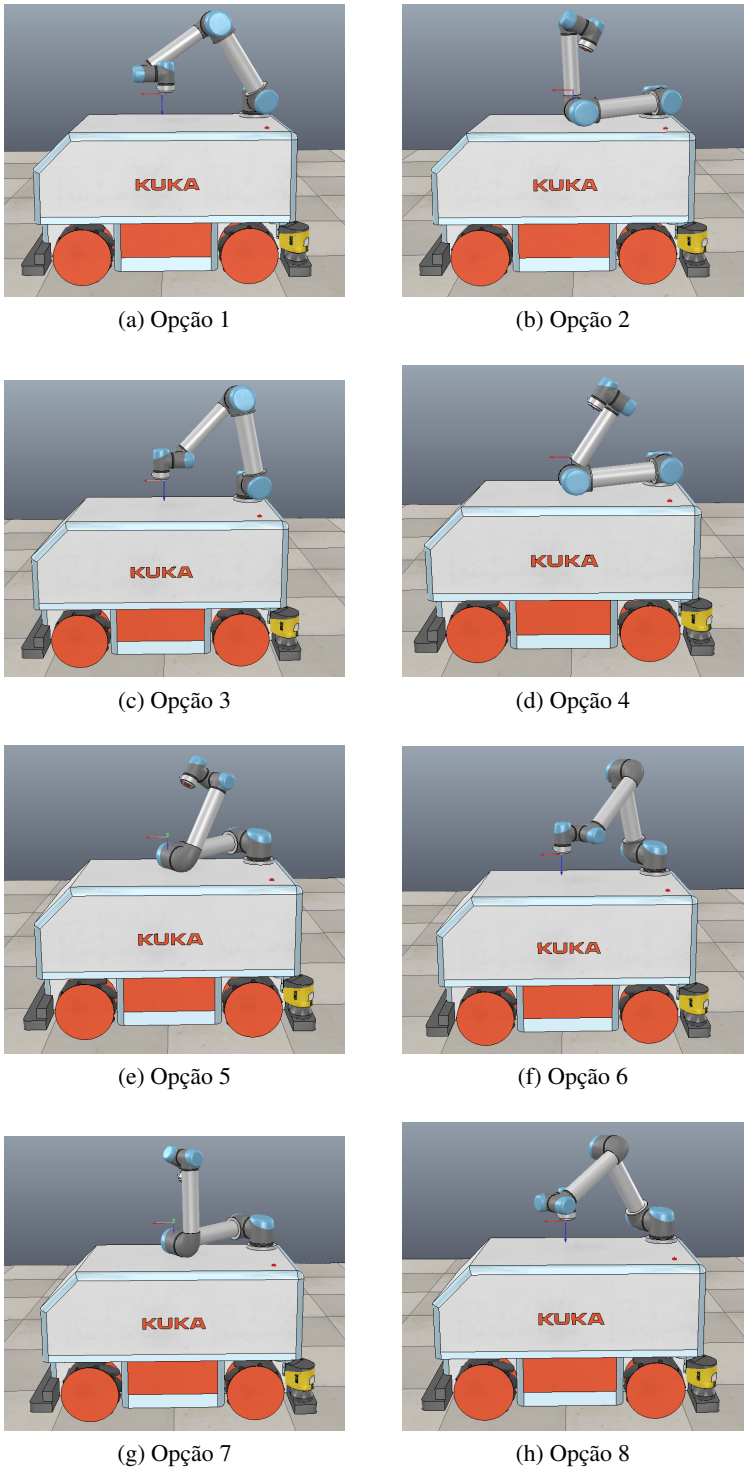
Como forma de teste, posicionou-se a plataforma na posição $x = 0$ e $y = 0$, seguidamente definiram-se as coordenadas do ponto de destino como sendo (0,0,0.8) e orientação segundo o eixo $-z$. Através da equação (4.2) obtém-se as coordenadas e a orientação do ponto em relação ao referencial do manipulador, aplicando a cinemática inversa obtém-se os 8 conjuntos de coordenadas (em radianos) correspondentes às várias configurações possíveis (tabela 4.1).

Tabela 4.1: Resultados da cinemática inversa.

Opção / Junta	1	2	3	4	5	6	7	8
θ_1	4,89015	4,89015	4,89015	4,89015	2,21861	2,21861	2,21861	2,21861
θ_2	5,28075	0,525628	4,92228	0,652062	2,48957	4,50252	2,61594	4,14401
θ_3	1,61149	4,67169	2,16324	4,11995	2,16323	4,11996	1,6115	4,67168
θ_4	0,96106	2,65598	3,90937	6,22289	3,20172	5,51522	0,485483	2,18041
θ_5	1,57095	1,57095	4,71223	4,71223	1,57035	1,57035	4,71284	4,71284
θ_6	1,74855	1,74855	4,89014	4,89014	5,3602	5,3602	2,21861	2,21861

Aplicando o algoritmo 2, obtêm-se que apenas as opções 1, 3, 6 e 8 são viáveis uma vez que as

restantes invadem o espaço definido pela plataforma. Esses resultados são comprovados na figura 4.10.



(a) Opção 1 (b) Opção 2

(c) Opção 3 (d) Opção 4

(e) Opção 5 (f) Opção 6

(g) Opção 7 (h) Opção 8

Figura 4.10: Resultado da cinemática inversa.

4.4 Controlo baseado em dois sistemas independentes

Nesta abordagem é desenvolvida uma lógica de controlo baseada no alcance do manipulador e na sua manipulabilidade. Definido um ponto no mapa, caso esse ponto seja inalcançável para o manipulador, então a plataforma deve aproximar-se do ponto até a distância euclidiana entre a base do manipulador e o ponto ficar dentro de um limite desejado tal que o ponto fique ao alcance do manipulador.

Posteriormente é resolvida a cinemática inversa do manipulador, através do algoritmo explicado no capítulo 3 e é escolhida a configuração que maximiza a manipulabilidade, tendo em conta um dos critérios definidos nas equações (2.11) e (2.12). Uma vez que esta abordagem efetua o

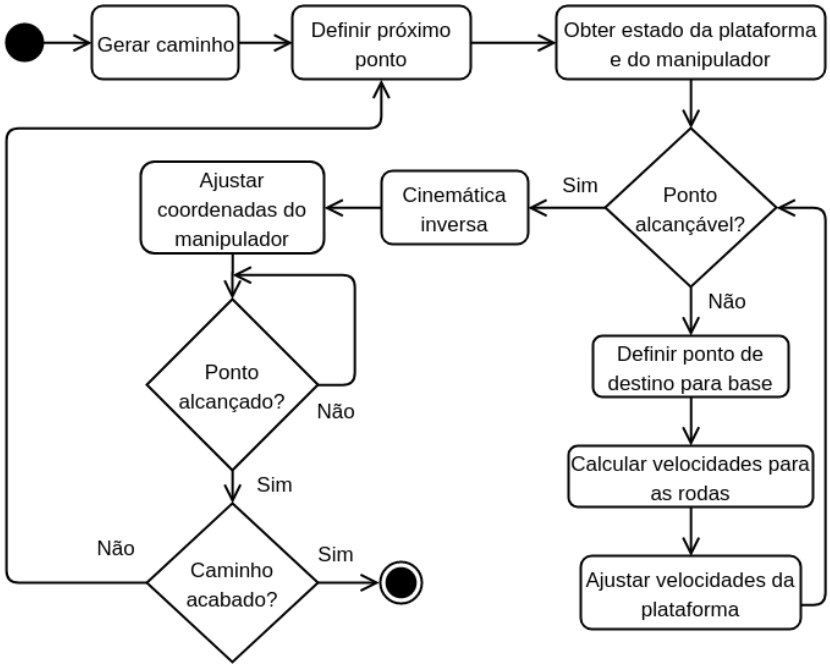


Figura 4.11: Diagrama de atividade para controlo dos dois subsistemas.

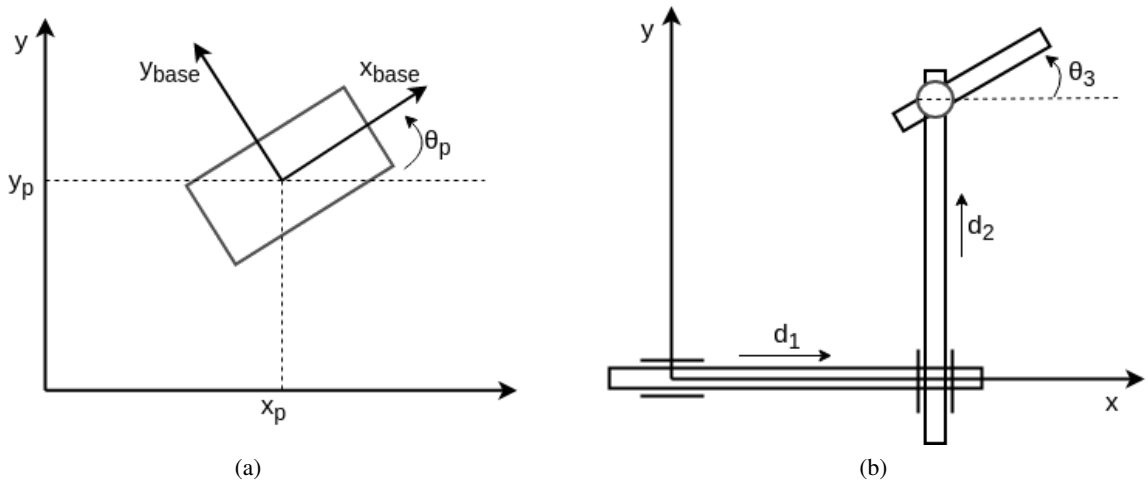
controlo da plataforma móvel e do manipulador de forma independente, torna-se mais simples. Na figura 4.11 pode ver-se um diagrama de atividade que exemplifica esta lógica de controlo, aplicada ao caso em que o manipulador tem de alcançar uma sequência de pontos no espaço que formam um caminho.

4.5 Controlo do sistema integrado

Devido à capacidade de movimentação da plataforma existem vários posicionamentos possíveis que permitem o manipulador atingir o ponto desejado. Contudo, segundo a abordagem explicada na secção 4.4, ao efetuar-se o controlo da plataforma móvel e do manipulador de forma independente tem-se a desvantagem de não se garantir o posicionamento ideal da plataforma, pois apenas se aproxima a plataforma do ponto de destino, podendo haver outros posicionamentos que

permitiriam ter uma manipulabilidade maior. Outra desvantagem de se fazer um controlo independente prende-se com o facto de apenas se mover um sistema de cada vez.

Para contornar isto pode-se aproximar a plataforma omnidirecional por uma cadeia cinemática de 3 juntas em que as duas primeiras são prismáticas e a última é do tipo rotativa. Desta forma, consegue-se reproduzir a posição da plataforma (x_p, y_p, θ_p) através das coordenadas das juntas do manipulador (d_1, d_2, θ_3), em que $x_p = d_1$, $y_p = d_2$ e $\theta_p = \theta_3$ (figura 4.12). Assim sendo, toda



(a) (b)

Figura 4.12: Aproximação da plataforma omnidirecional por um manipulador.

a cadeia cinemática pode ser estudada como sendo um manipulador de 9 juntas, que resulta da integração do manipulador que modela a plataforma omnidirecional com o manipulador UR5.

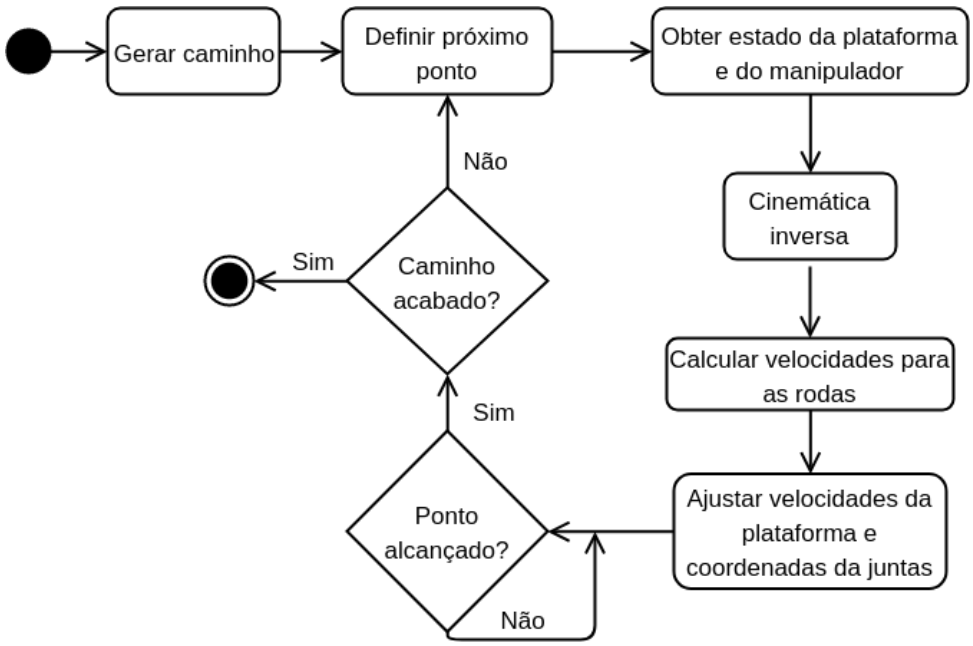


Figura 4.13: Diagrama de atividade para controlo do sistema.

A figura 4.13 mostra a lógica implementada para controlar o sistema.

4.5.1 Biblioteca *TRAC-IK*

Uma alternativa para a resolução da cinemática inversa, passa por usar bibliotecas que a resolvem. Como referido em [34], seguramente uma das mais usadas na robótica é a *Orocos Kinematics and Dynamics Library* (*KDL*)[2], que se baseia no método de *Newton* para encontrar solução, contudo esta apresenta alguns problemas, como:

- Frequentes falhas de convergência na presença em robôs com juntas limitadas;
- Por vezes o algoritmo fica "preso" em um mínimo local.

Deste modo, ao longo desta dissertação é usada a biblioteca *trac-ik*[3] que fornece métodos alternativos aos existentes na *KDL*. Segundo o autor de [34], para resolução da cinemática inversa esta biblioteca possui dois métodos, o primeiro é um simples melhoramento do método de *Newton* usado no *KDL*, que deteta e mitiga os mínimos locais quando os limites das juntas são encontrados, enquanto o segundo é uma otimização PQS (programação quadrática sequencial) que usa métodos *quasi-Newton* para lidar melhor com os limites das juntas. Por defeito, o *trac-ik* retorna imediatamente quando um desses métodos converge para uma resposta, contudo restrições como a distância ou manipulabilidade podem ser fornecidas ao algoritmo de maneira a obter-se a melhor solução.

Para utilizar esta biblioteca é necessário recorrer a ficheiros *URDF*[4] (*Unified Robot Description Format*) que se tratam de uma especificação em *XML* para descrever o modelo do robô. Esta especificação assume que o robô é constituído por um conjunto de elos rígidos ligados através de juntas (figura 4.14).

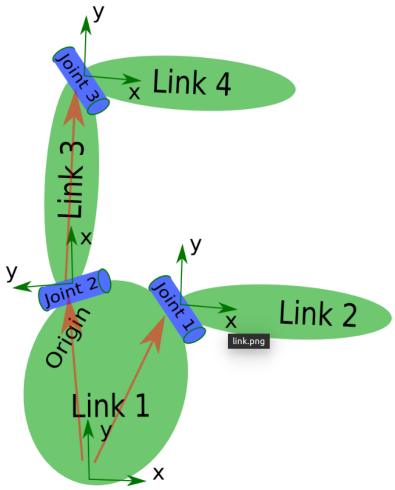
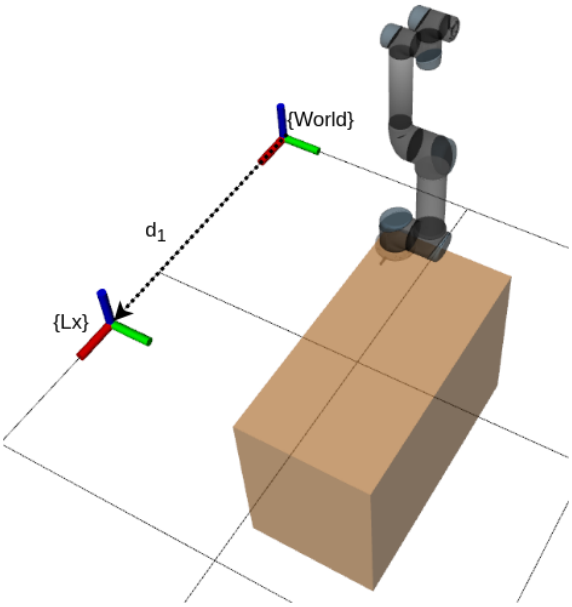


Figura 4.14: Especificação de um robô através de elos e juntas.

[2]http://www.orocos.org/wiki/orocos/kdl-wiki/.
[3]https://bitbucket.org/traclabs/trac_ik
[4]http://wiki.ros.org/urdf

A especificação *XML* do manipulador *UR5* é disponibilizado no pacote *ROS Universal_robot*[5]. Através dessa especificação acrescenta-se as juntas e elos necessários para formar o sistema desejado de 9 juntas (figuras 4.15, 4.16, 4.17, 4.18).



(a)

```xml
<?xml version="1.0"?>

    <link name ="World">
    </link>

    <joint name="d1" type="prismatic">
    <parent link="World"/>
    <child link="Lx"/>
    <axis xyz="1 0 0"/>
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <limit effort="30" lower="-3.5"
    upper="3.5" velocity="0.5"/>
    </joint>

    <link name="Lx">
    </link>
```

(b)

Figura 4.15: Desenvolvimento da junta prismática em x.

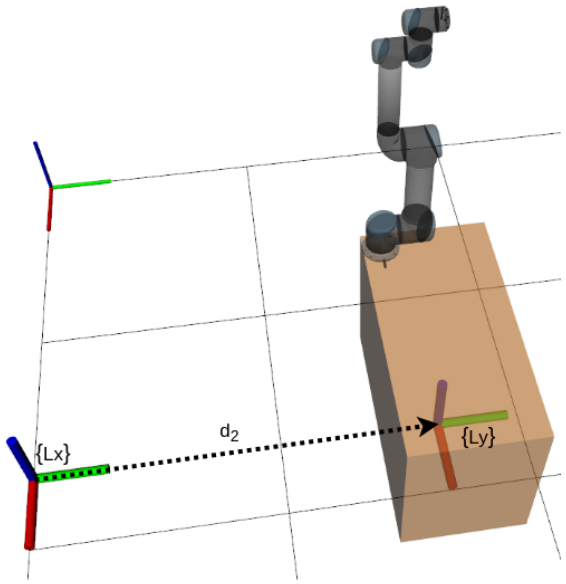
Na figura 4.15 tem-se a especificação da junta prismática que demonstra a capacidade de movimentação da plataforma ao longo do eixo x, essa junta faz a ligação entre os elos "World" e "Lx". Os limites da junta são definidos em metros, uma vez que se trata de uma junta prismática. Na figura 4.16 pode ver-se o mesmo processo feito para descrever o movimento em y.

A componente rotacional da plataforma está descrita na figura 4.17. Neste caso, uma vez que a plataforma pode rodar continuamente em torno do seu eixo, representa-se através de uma junta do tipo "*continuous*". Este tipo de juntas é adequada para representar rotações onde não existem limites.

Este conjunto de elos e juntas, que formam a cadeia cinemática que representa a plataforma, são então anexados à base do manipulador através de uma junta do tipo "*fixed*"(figura 4.18). Este tipo de junta é assim denominada pois acaba por não ser uma verdadeira junta, visto que esta não se pode mover uma vez que tem todos os graus de liberdade bloqueados. O restante do ficheiro *XML* manteve-se inalterado. Tendo o ficheiro *URDF* que descreve a cadeia cinemática total, pode então utilizar-se a biblioteca *trac-ik* para resolver a cinemática da mesma. É importante salientar que até ao momento esta biblioteca não garante que a solução da cinemática inversa não resulte em auto-colisões, portanto é necessário validar a solução através do algoritmo 2.

Como referido anteriormente, é possível escolher restrições de modo a se obter a melhor solução da cinemática inversa. Uma dessas restrições é a manipulabilidade. Tendo em conta as

[5]http://wiki.ros.org/universal_robot

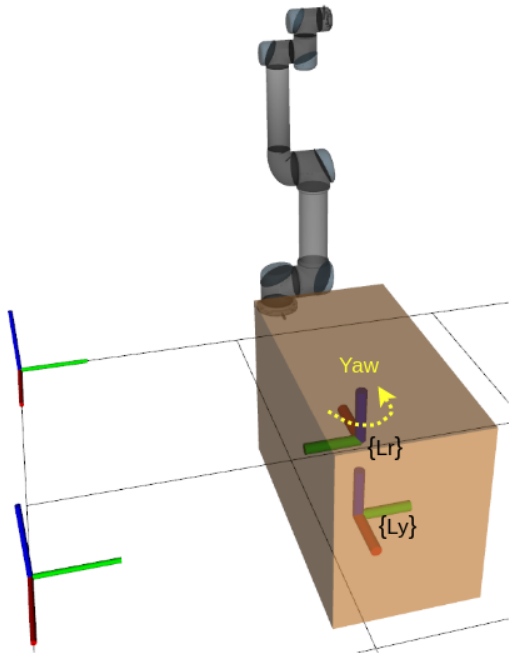


(a)

```
15    <link name="Lx">
16    </link>
17
18  <joint name="d2" type="prismatic">
19    <parent link="Lx"/>
20    <child link="Ly"/>
21    <axis xyz="0 1 0"/>
22    <origin rpy="0 0 0" xyz="0 0 0"/>
23    <limit effort="30" lower="-3.5"
24    upper="3.5" velocity="0.3"/>
25  </joint>
26
27    <link name="Ly">
28    </link>
```

(b)

Figura 4.16: Desenvolvimento da junta prismática em *y*.



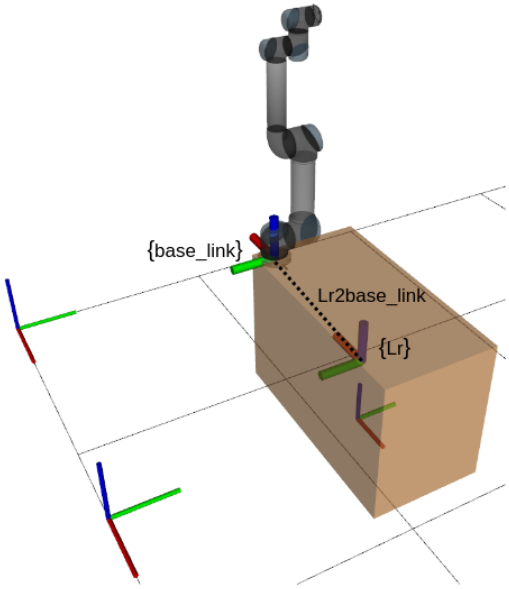
(a)

```
27    <link name="Ly">
28    </link>
29
30    <joint name="rotation" type="continuous">
31    <parent link="Ly"/>
32    <child link="Lr"/>
33    <axis xyz="0 0 1"/>
34    <origin rpy="0 0 3.1415" xyz="0 0 0.32975"/>
35    </joint>
36
37  <link name="Lr">
38  </link>
```

(b)

Figura 4.17: Desenvolvimento da junta rotacional.

equações (2.11) e (2.12) uma vez que se pretendendo maximizar a manipulabilidade do manipulador, ao invés da manipulabilidade da cadeia total, alterou-se a biblioteca original de forma a ser calculada apenas a matriz jacobiana do mesmo. Consequentemente, obtém-se o cálculo correto da manipulabilidade.



(a)

```
37  <link name="Lr">
38  </link>
39
40   <joint name="lr2base_link" type="fixed">
41   <parent link="Lr"/>
42   <child link="${prefix}base_link"/>
43   <origin rpy="0 0 0" xyz="0.43 0.18 0.32975"/>
44   </joint>
```

(b)

Figura 4.18: Ligação da base do UR5 à plataforma móvel.

Como forma de teste à biblioteca e ao ficheiro *URDF* desenvolvido, definiu-se a matriz homogênea T_{ponto}^{mundo} e utilizou-se a biblioteca *trac-ik* para obter as coordenadas para cada junta do sistema.

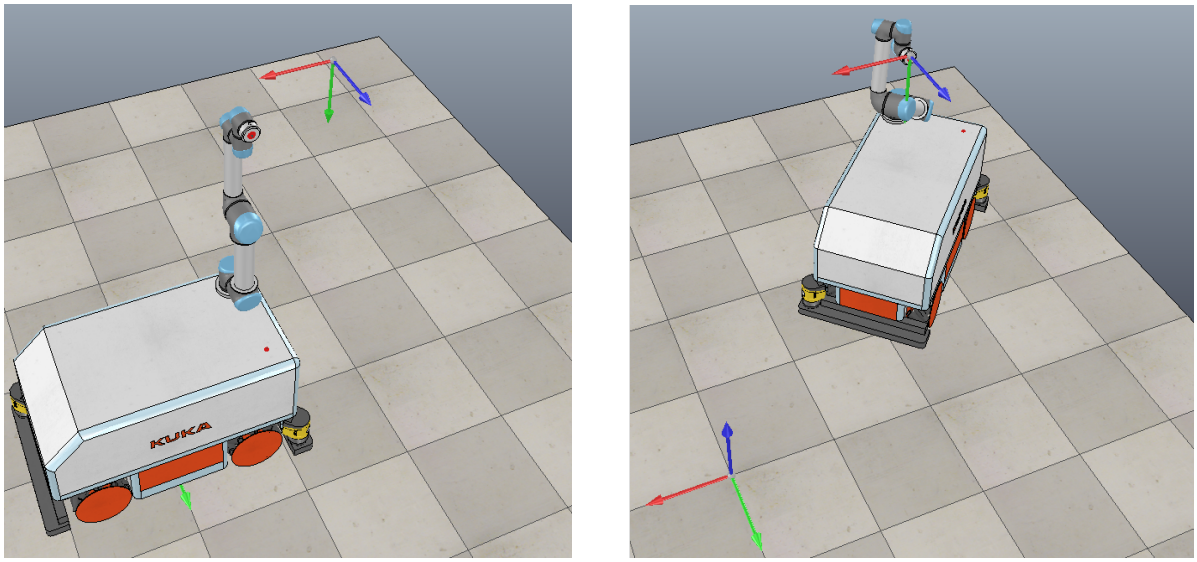
$$T_{ponto}^{mundo} = \begin{bmatrix} 1 & 0 & 0 & -1,3 \\ 0 & 0 & 1 & -1 \\ 0 & -1 & 0 & 1,287 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.7}$$

A tabela 4.2 mostra os resultados obtidos para o ponto desejado. Como explicado anteriormente as 3 primeiras juntas referem-se à deslocação e orientação da plataforma enquanto as restantes 6 juntas correspondem aos valores das coordenadas das juntas do manipulador.

Tabela 4.2: Resultado para o ponto definido

	Junta 0	Junta 1	Junta 2	Junta 3	Junta 4	Junta 5	Junta 6	Junta 7	Junta 8
Resultado	-1,2841	-0,9235	0,8117	-0,0900	0,9585	-1,0926	-3,0074	0,7209	3,1415

Através do *V-REP* obtém-se o resultado demonstrado na figura 4.19. Partindo da posição inicial (figura 4.19a) a plataforma moveu-se para a posição $(x,y) = (-1,2841, -0,9235)$ e ficou com uma orientação de 0,8117 radianos ($\approx 46,5^{o}$) enquanto o manipulador moveu as suas juntas para as coordenadas desejadas fazendo a ferramenta de trabalho alcançar o ponto desejado com a orientação correta (figura 4.19b).



(a) Posição inicial (b) Posição final

Figura 4.19: Posição inicial e final.

4.6 Resultados

Para se testar as duas abordagens propostas, definiu-se um conjunto de pontos de destino para a ferramenta de trabalho. A figura 4.20 mostra o ambiente de simulação assim como a trajetória utilizada no primeiro teste.

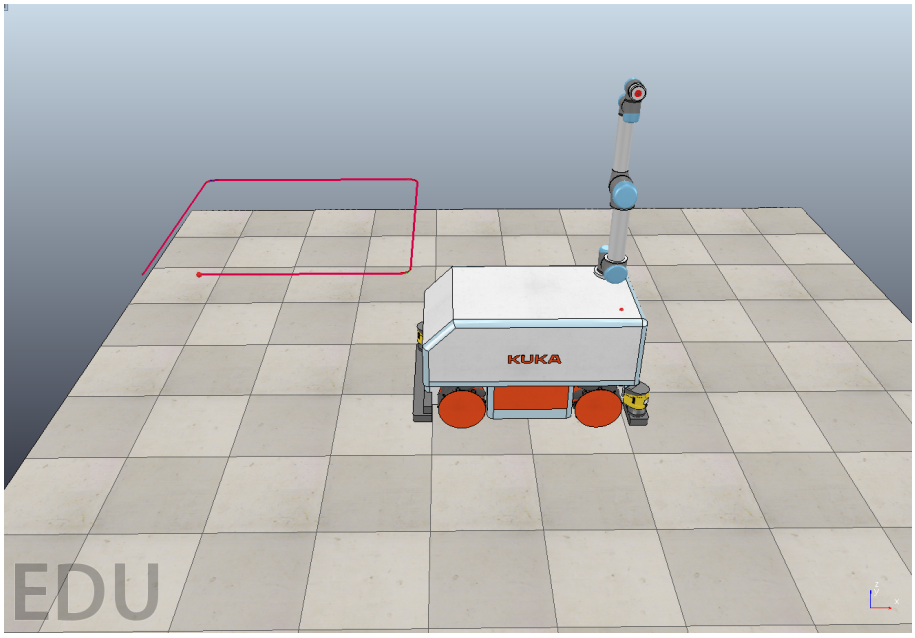


Figura 4.20: Ambiente da primeira simulação.

A figura 4.21 mostra os resultados para a primeira abordagem em que o controlo do manipulador e da plataforma é feito de forma independente. Os pontos vermelhos representam as posições

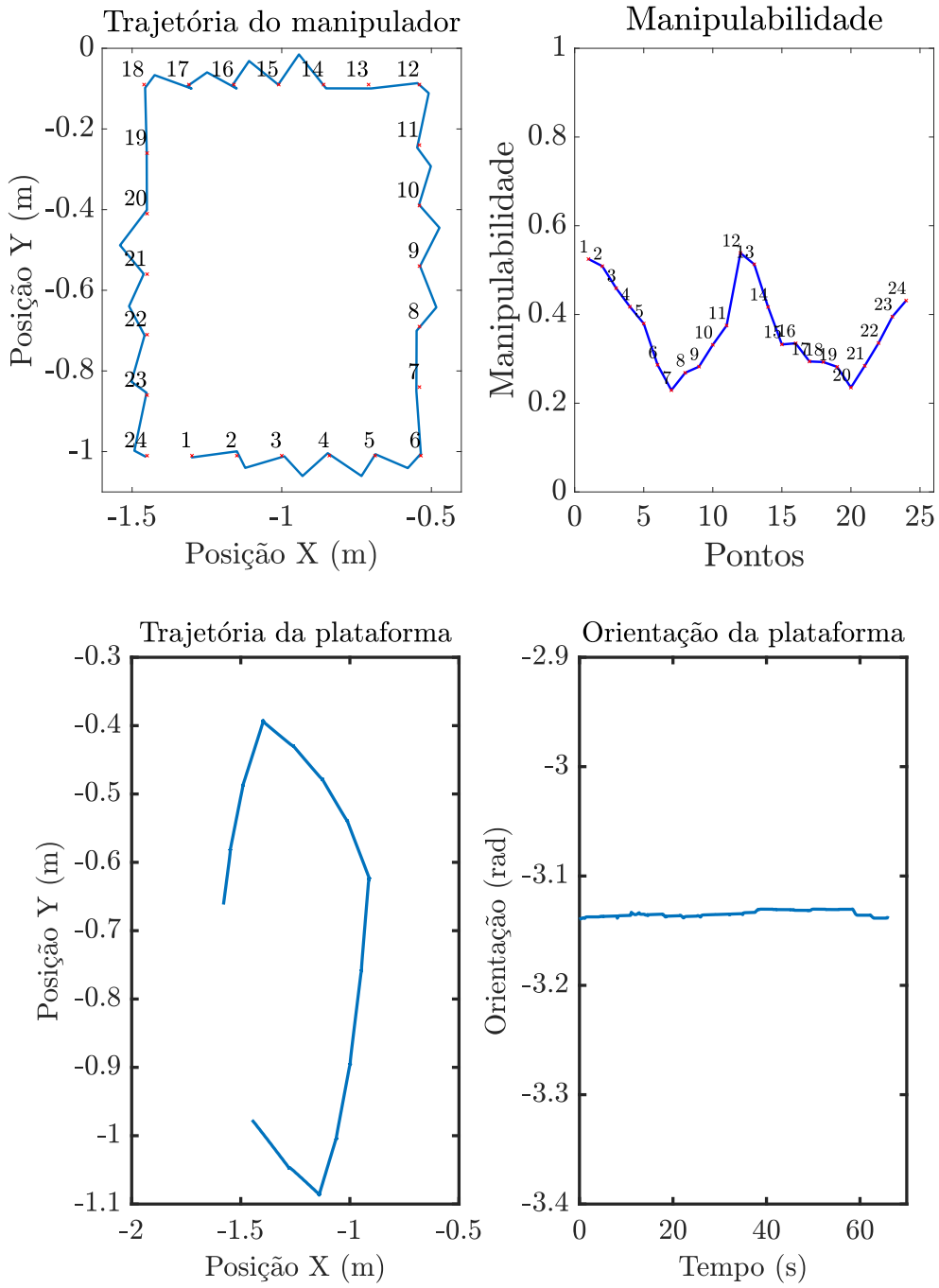


Figura 4.21: Resultados da primeira abordagem para trajetória retangular.

que a ferramenta de trabalho deve alcançar. O ponto 1 e 24 são os pontos de início e fim da trajetória, respetivamente.

Esta abordagem resulta numa trajetória irregular por parte do manipulador, algo que já era esperado, uma vez que sempre que um ponto está fora do alcance do manipulador, a plataforma move-se até perto deste e só depois se move o manipulador. Nos casos em que o próximo ponto está dentro do alcance do manipulador, uma vez que apenas o manipulador tem de se mover, a

trajetória efetuada entre os dois pontos é menos irregular. Os pontos 18 e 19, por exemplo, são casos que encaixam nesta situação. Uma vez que existem grandes variações no movimento do manipulador, devido ao facto de este ajustar a sua posição sempre que a plataforma se mexe, existem também grandes variações na manipulabilidade, como se pode ver no gráfico referente a esta.

A figura 4.22 mostra os resultados para a segunda abordagem. O caminho definido nesta abordagem foi o mesmo da primeira abordagem. Neste caso, como o manipulador e a plataforma se movem em conjunto quando um ponto está fora do alcance do manipulador, pode ver-se que a ferramenta de trabalho viaja entre pontos consecutivos de forma bastante mais regular, comparativamente à abordagem anterior. Pode ver-se também que a manipulabilidade no início da trajetória (ponto 1), relativamente à abordagem anterior, é maior. Isto acontece devido ao facto da cinemática inversa ter em conta o sistema total, escolhendo a melhor posição para posicionar a plataforma que maximiza a manipulabilidade do braço. Uma vez que o manipulador apresenta uma trajetória menos irregular, a sua manipulabilidade também apresenta significativamente menos variações.

Analisando os dados referentes à plataforma, percebe-se que em ambos os casos existem 3 zonas em que a plataforma muda bruscamente de direção. Isso já era esperado, uma vez que a trajetória definida para o manipulador também o faz. É ainda possível ver-se que a segunda abordagem produziu uma trajetória para a plataforma mais suave, ao passo que na primeira a plataforma muda ligeiramente de direção de cada vez que o manipulador tem de alcançar um novo ponto. Estas trajetória mais suave também acontece devido ao facto de na segunda abordagem a plataforma poder mudar a sua orientação ao longo da trajetória. A primeira abordagem não lida com a orientação do robô, sendo esta a mesma ao longo de toda a trajetória.

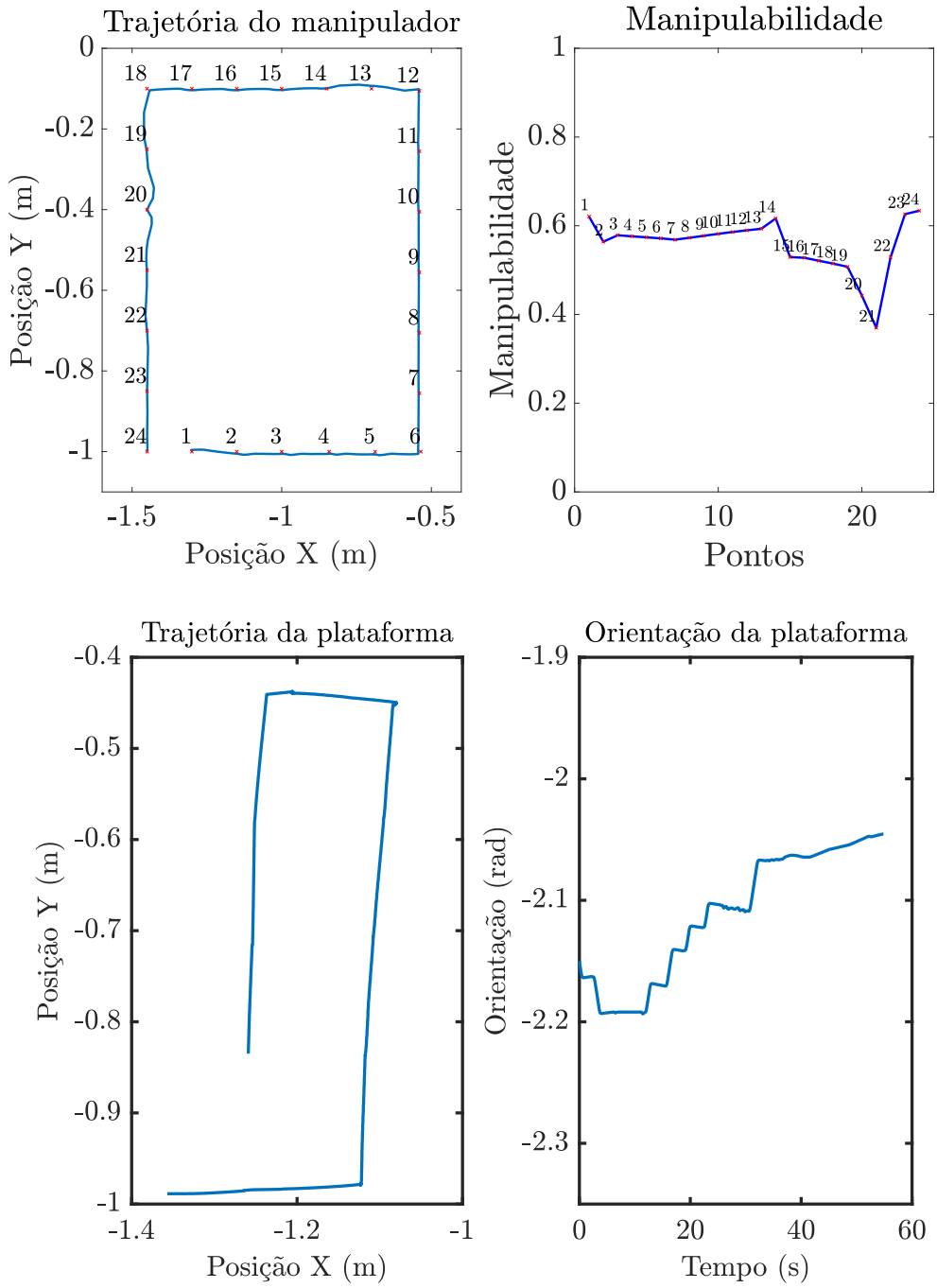


Figura 4.22: Resultados da segunda abordagem para trajetória retangular.

Efetuou-se ainda um segundo teste, no qual se definiu uma trajetória sinusoidal ao longo do plano *xz*. Na figura 4.23 pode ver-se o ambiente de simulação utilizado nesta segunda abordagem.

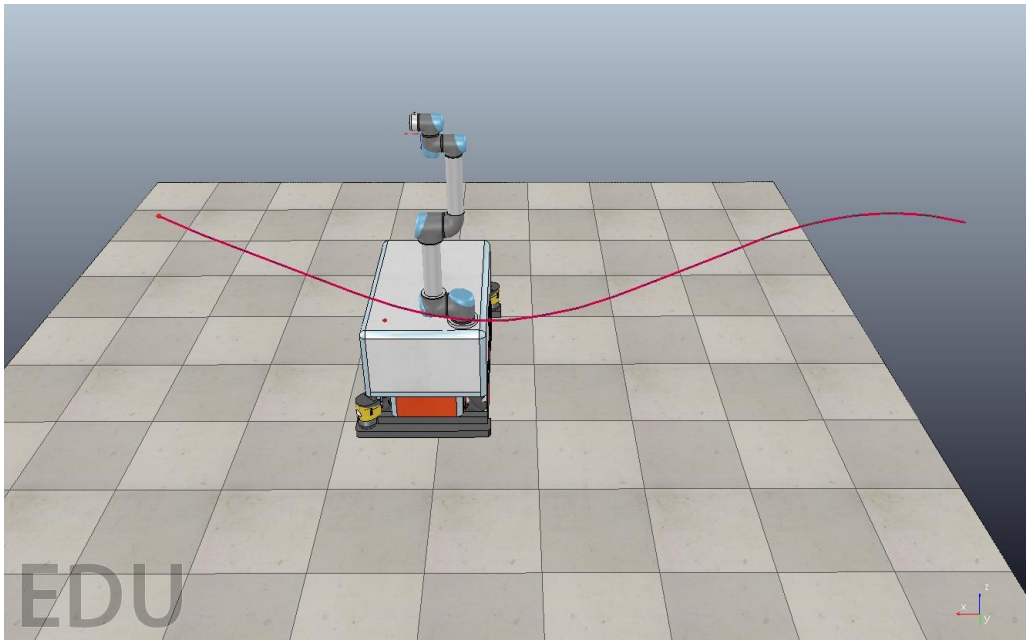


Figura 4.23: Ambiente da segunda simulação.

A figura 4.24 mostra os resultados obtidos para a primeira abordagem. Assim como no exemplo anterior, esta abordagem resulta numa trajetória para o manipulador mais irregular quando comparado com o resultado da segunda abordagem (figura 4.25), devido aos motivos já explicados no teste anterior. Relativamente à manipulabilidade, pode ver-se novamente que a segunda abordagem permite ter valores superiores aos obtidos com a primeira abordagem.

Relativamente às trajetórias da plataforma, na primeira abordagem esta aproximou-se gradualmente do plano onde estava definida a sinusoide. A partir desse momento, uma vez que a sinusoide se move apenas ao longo desse plano, a plataforma moveu-se também, praticamente em linha reta, ao longo do mesmo. Na segunda abordagem, a plataforma foi se movendo e afastando do plano formado pela sinusoide, fazendo uso novamente da capacidade de rotação da plataforma.

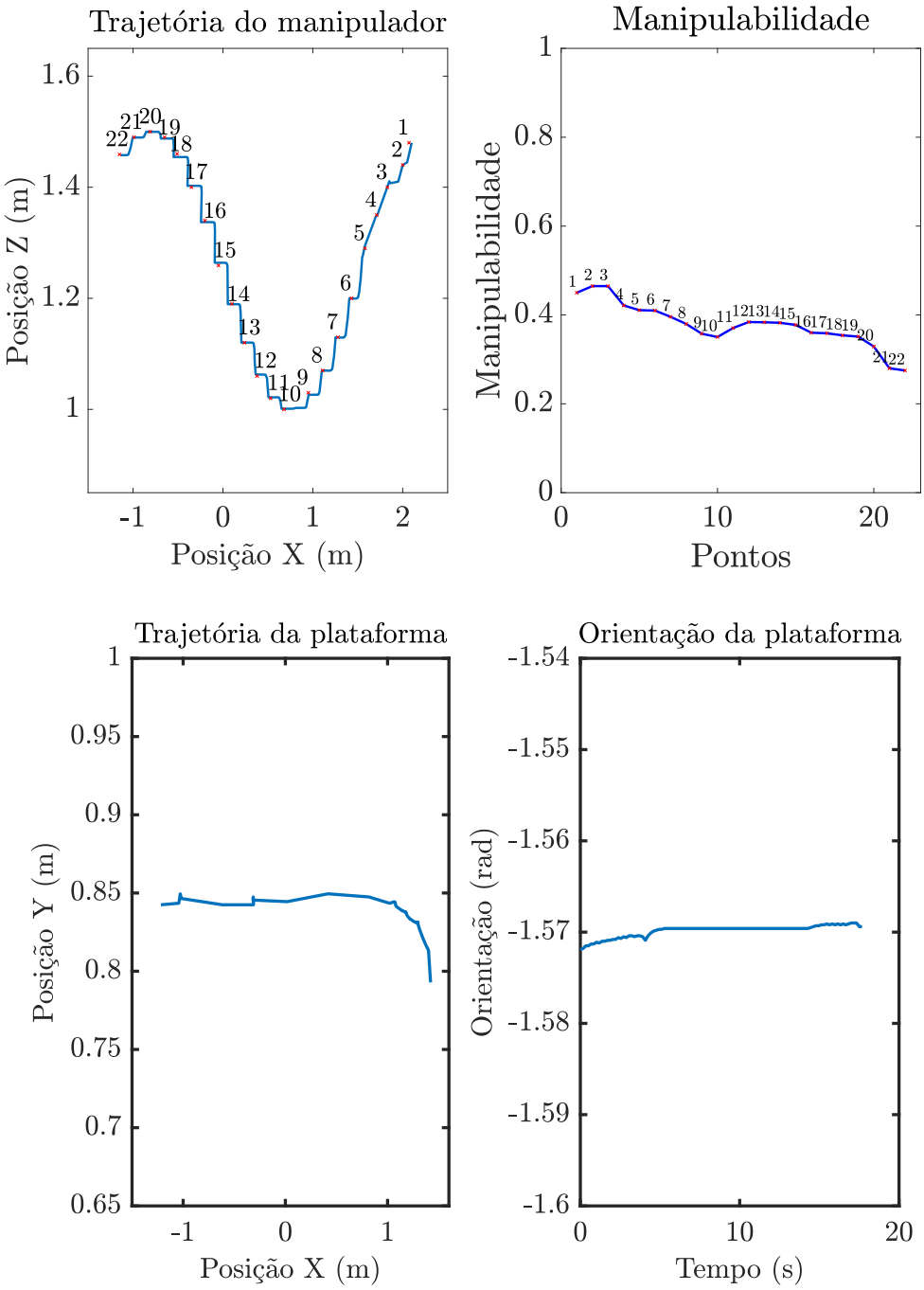


Figura 4.24: Resultados da primeira abordagem para trajetória sinusoidal.

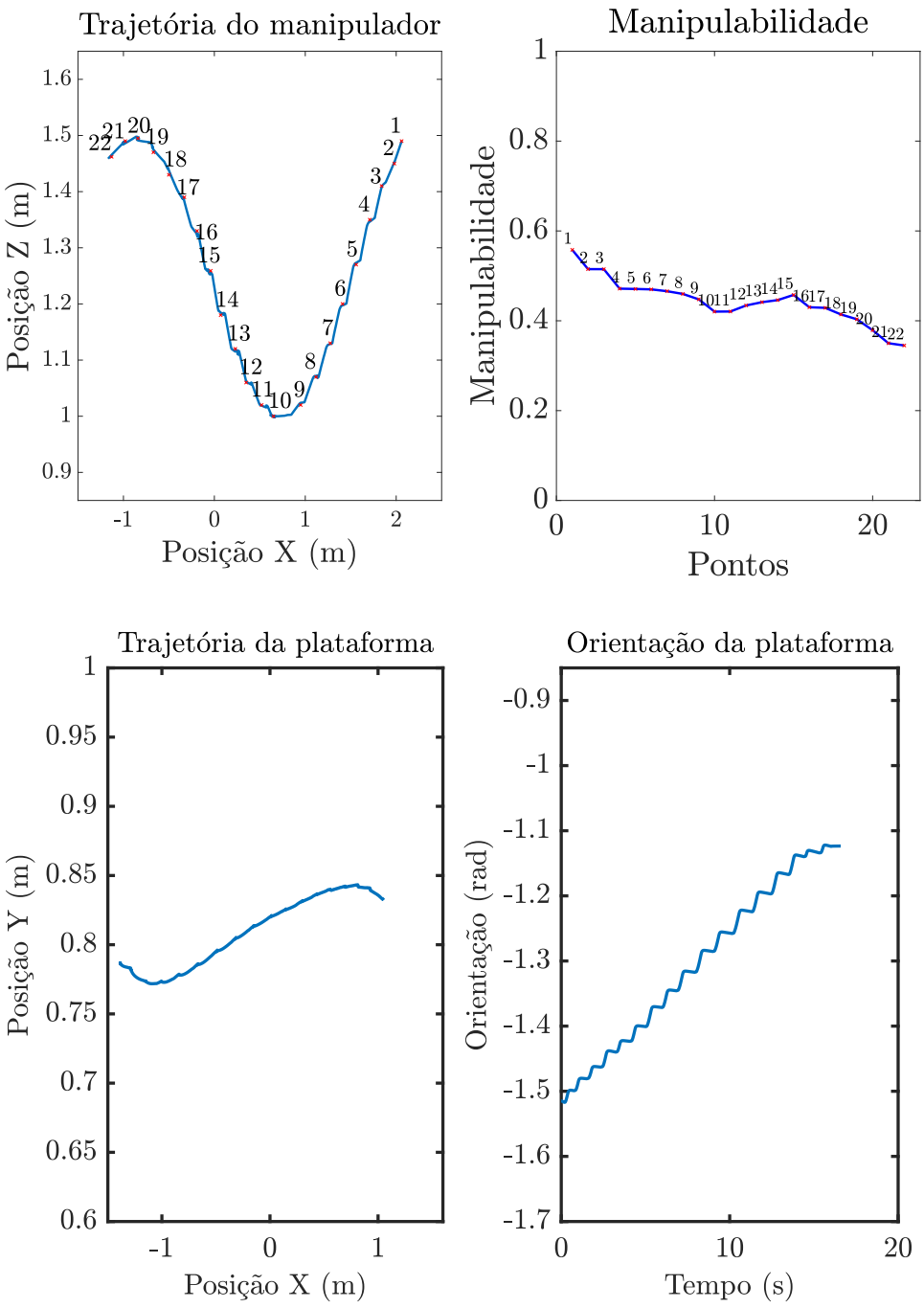


Figura 4.25: Resultados da segunda abordagem para trajetória sinusoidal.

4.7 Conclusão

Neste capítulo inicialmente fez-se uma explicação das ferramentas utilizadas para o desenvolvimento desta dissertação. Posteriormente desenvolveu-se e testou-se o controlo do manipulador e da plataforma móvel. Relativamente ao controlo do manipulador móvel, desenvolveu-se duas abordagens. A primeira delas tratou o manipulador e a plataforma móvel de forma independente, enquanto na segunda abordagem se fez a integração destes dois sistemas, com vista a resolver-se a cinemática composta do sistema. Os resultados mostram que a segunda abordagem gera trajetórias mais suaves para o manipulador.

A resolução da cinemática do sistema integrado (modelado por um manipulador de 9 juntas), mostrou bons resultados quando se analisa a trajetória do manipulador. Contudo, na presença de obstáculos podem surgir problemas devido ao facto desta abordagem permitir à plataforma móvel viajar livremente pelo plano *xy*. O capítulo 5 lida com esse problema.

Capítulo 5

Planeamento de trajetórias

O planeamento de trajetórias evitando colisões é um problema típico onde a coordenação de todo o sistema é crucial. Para se mover a ferramenta de trabalho desde o ponto A até ao ponto B é necessário criar um caminho ótimo que una esses pontos evitando colisões com obstáculos. Os algoritmos de planeamento de trajetórias são responsáveis por gerar esse caminho.

No caso particular dos manipuladores móveis, estes são utilizados em muitas ocasiões para facilitar a movimentação da ferramenta de trabalho ao longo de um determinado caminho. Desta forma também é necessário gerar uma trajetória para a plataforma, de modo a que esta evite obstáculos. Além disso, é necessário garantir que a trajetória gerada para a plataforma não afete a trajetória da ferramenta de trabalho.

Para tal implementou-se a seguinte lógica. Inicialmente são definidos um ponto inicial e um ponto de destino para a ferramenta de trabalho. Com estes dois pontos é possível obter-se uma trajetória ótima. Depois de obtida esta trajetória, gera-se uma outra para a plataforma, dependente da primeira, a qual deve garantir que a ferramenta de trabalho consegue realizar a sua trajetória. Para cumprir esta restrição, todos os pontos que não cumprem esta condição (ou seja, todos os pontos demasiado distantes dos pontos que formam a trajetória do manipulador) são definidos como obstáculos. Desta forma, garante-se que a trajetória gerada para a plataforma apenas passa por pontos do mapa que permitem ao manipulador alcançar a sua trajetória.

Em suma, a abordagem implementada segue os seguintes passos:

1. Definir ponto inicial e ponto final para o manipulador;
2. Calcular trajetória ótima para o manipulador;
3. Calcular posição inicial e final para a plataforma;
4. Definir zona interdita do mapa;
5. Calcular trajetória ótima para a plataforma.

Para o cálculo do caminho ótimo é usado o algoritmo A*, o qual vai ser explicado em pormenor na secção 5.1.

5.1 Algoritmo A*

Como referido em 2.2.2.1 o algoritmo A* é um algoritmo de procura em grafo responsável por encontrar o caminho mais curto entre um nó inicial e um nó de destino. Outra das suas características é o facto de efetuar pesquisas ótimas, ou seja, é um algoritmo capaz de determinar a sequência de nós que minimiza uma função de custo, determinada pelos custos associados a cada ligação entre nós. Esses custos dependem do fator que se pretende otimizar. No caso desta dissertação, esse fator é a distância.

Desta forma, para implementar este algoritmo o primeiro passo é fazer uma divisão do mapa em células de forma a obter um conjunto de nós para o A* pesquisar. Esta divisão de células pode seguir várias abordagens consoante o nível de complexidade e precisão que se necessita. De forma a reduzir a complexidade computacional, optou-se por se utilizar uma composição aproximada por células fixas, com forma geométrica quadrada. Assim sendo, dividiu-se o mapa em 20 células de largura, 20 de comprimento e 10 de altura. Posteriormente, cada célula é considerada livre ou ocupada, de acordo com análise do mapa.

Cada célula representa um nó e tem ligação direta às suas células adjacentes. Estas adjacências serão determinadas através de uma pesquisa de conectividade 4, isto significa que qualquer célula que partilhe uma aresta com a célula atual, é considerada adjacência. O custo dessa ligação é dado pela distância entre as duas células, sendo geralmente esse valor igual a 1.

Para determinar qual a ordem pela qual se deve percorrer os nós de forma a encontrar, o mais rapidamente possível, o caminho com custo mínimo entre o nó inicial e o nó de destino, este algoritmo utiliza uma função heurística $f(n)$. Esta função resulta da soma de outras duas funções (figura 5.1):

- $g(n)$ representa o custo desde o nó inicial até ao nó n. Pode ou não ser uma função heurística.
- $h(n)$ representa o custo estimado desde o nó n até ao nó de destino. É uma função heurística.

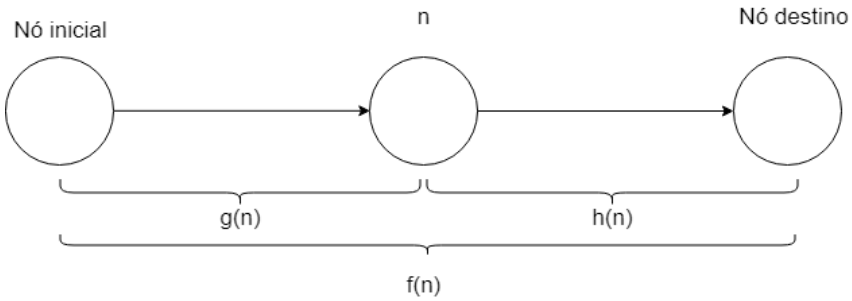


Figura 5.1: Função $f(n)$, $g(n)$ e $h(n)$.

Existem várias heurísticas possíveis a usar num mapa de células. Convém usar heurísticas simples e o mais reias possíveis, para se obter melhores os resultados. A heurística implementada foi a distância *Manhattan*, na qual se calcula a distância em três direções, uma vez que se está a trabalhar no plano 3D. Relativamente à notação usada, D representa o peso de mudança de célula

e pode ser considerado 1, n representa o nó para o qual se está a calcular a heurística e *destino* representa a célula de destino.

$$h(n) = D \times (abs(n_x - destino_x) + abs(n_y - destino_y) + abs(n_z - destino_z)) \tag{5.1}$$

A figura 5.2 apresenta um exemplo da heurística de *Manhattan* no plano 2D, nesse exemplo $h(n) = 2+3 = 5$.

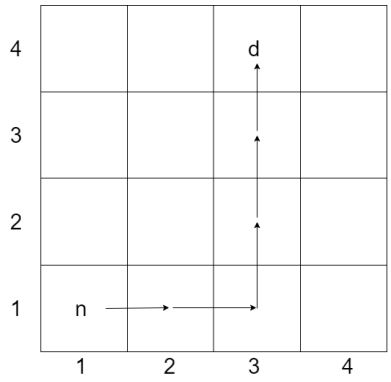


Figura 5.2: Heurística distância de *Manhattan*.

O algoritmo 3 descreve o funcionamento do A*. Na execução deste algoritmo são utilizadas duas listas: uma lista aberta ($Lista_{aberta}$) para os nós visitados mas que ainda não foram expandidos e uma lista fechada ($Lista_{fechada}$) para os nós que já foram processados, ou seja, para os nós que já saíram da lista aberta. É ainda usada uma função $c(n_1, n_2)$ que representa o custo de passar do nó n_1 para o nó n_2. Também são utilizadas as funções $detetaColisao(n)$ e $Heuristica(n)$. A primeira função retorna 1 caso o nó n seja considerado um obstáculo, enquanto a segunda retorna o valor da equação (5.1). No que diz respeito à terminologia usada que ainda não foi explicada, $N_{adjacente}$, N_{atual}, $N_{pai}(N_{adjacente})$ representam o nó adjacente, nó atual e o nó que deu origem ao nó adjacente, respetivamente.

Depois de executado o algoritmo, a melhor trajetória é obtida percorrendo o nó pai, sucessivamente, desde o nó destino até se chegar ao nó inicial. No caso em que a lista aberta fique vazia sem que o nó de destino seja alcançado, significa que não existe caminho entre os dois nós.

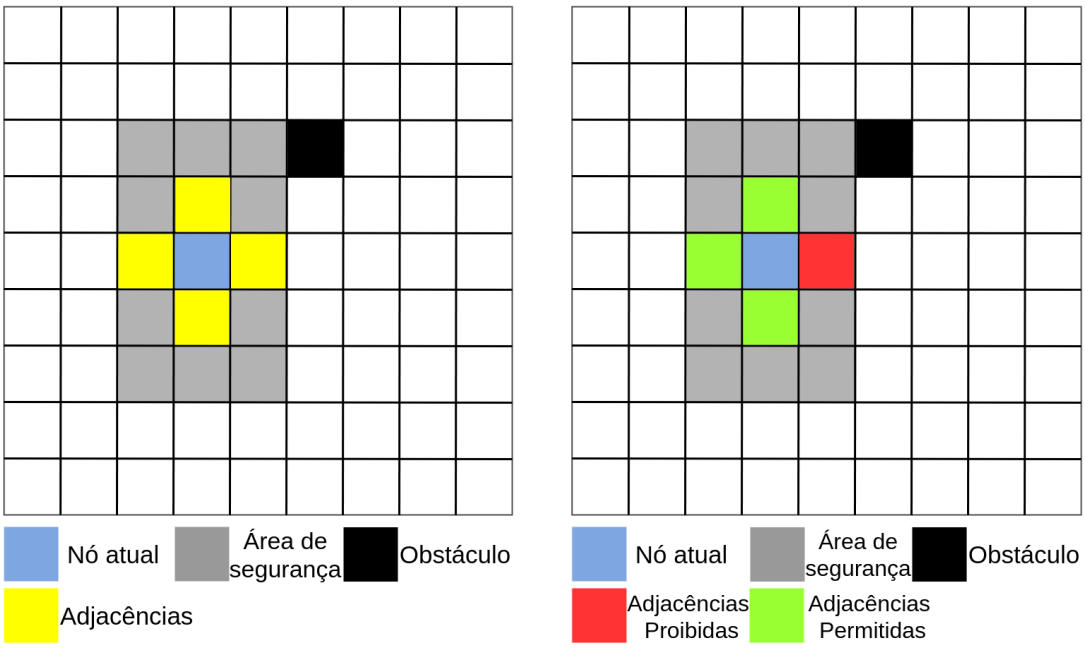
Algoritmo 3: A*

```
1  Inserir nó inicial na Lista_aberta ;
2  while Lista_aberta não estiver vazia do
3      Escolher o nó da Lista_aberta com menor função de custo f(n);
4      Retirar o melhor nó da Lista_aberta (N_atual) e adicionar à Lista_fechada;
5      if N_atual = N_destino then
6          Terminar algoritmo;
7      end
8      Gerar todos os nós adjacentes de N_atual;
9      foreach N_adjacente de N_atual do
10         if detetaColisao(N_adjacente) OR N_adjacente ∈ Lista_fechada then
11             continue;
12         end
13         if N_adjacente ∉ Lista_aberta then
14             N_pai(N_adjacente) = N_atual;
15             g(N_adjacente) = g(N_atual) + c(N_atual, N_adjacente);
16             h(N_adjacente) = heuristica(N_atual);
17             Insere N_adjacente na Lista_aberta;
18         end
19         else if g(N_atual) + c(N_atual, N_adjacente) < g(N_adjacente) then
20             N_pai(N_adjacente) = N_atual;
21             g(N_adjacente) = g(N_atual) + c(N_atual, N_adjacente);
22     end
23 end
```

Visto que o algoritmo desenvolvido será usado para planear tanto as trajetórias do manipulador como da plataforma, é necessário ter em consideração que o manipulador se pode mover em 3 dimensões, contudo, a plataforma apenas o pode fazer no plano xy. Desta forma, quando é executado o planeamento da trajetória para a plataforma, deve considerar-se a altura do mapa igual a 1, de maneira a garantir uma solução fazível.

Uma vez que o A* é um algoritmo de pesquisa em grafo, a plataforma móvel é considerada do tamanho da célula durante a execução do algoritmo. Contudo, esta ocupa várias células devido às suas dimensões. Deste modo, para evitar colisões, é necessário criar uma área de segurança em torno da célula, de acordo com as dimensões e orientação da plataforma. Uma vez que a plataforma em causa ocupa 3 células de largura e 5 de comprimento, desta forma, quando se verifica se determinada adjacência é ou não obstáculo, tem de se garantir que à sua volta exista uma margem de segurança com dimensões idênticas às do robô.

Na figura 5.3a estão representados o nó atual e a zona de segurança em torno dele. Estão ainda representadas as adjacências desse nó, resultantes da conectividade 4. Seguindo o algoritmo tradicional do A*, qualquer uma das adjacências do nó atual seriam consideradas células livres e posteriormente inseridas na lista aberta. Contudo, apesar das adjacências não serem obstáculos, é também necessário garantir que a zona de segurança em torno de cada uma delas é cumprida. Apenas se esta condição for cumprida é que se pode colocar a adjacência na lista aberta.



(a) Nó atual e adjacências.

(b) Adjacências proibidas e permitidas.

Figura 5.3: Zona de Segurança.

Na figura 5.3b pode ver-se a divisão das adjacências em "permitidas" e "proibidas". Esta divisão é o resultado da aplicação da condição anterior. Caso o robô se movesse para adjacência vermelha, a área de segurança iria invadir a célula do obstáculo, o que iria resultar numa colisão. Deste modo, essa adjacência deve ser considerada proibida, ou seja, descartada.

5.1.1 Aplicação na plataforma e no manipulador

O algoritmo foi testado tanto para gerar trajetórias para a plataforma como para o manipulador. No caso da plataforma, foram testados dois casos que apenas diferem na orientação da plataforma, as restantes condições (posição inicial, posição de destino e obstáculos) mantêm-se. Desta forma, consegue-se perceber a implicação da área de segurança no resultado final.

Considerando as figuras 5.4 e 5.5 como sendo o caso 1 e 2, respetivamente, pode-se observar que as condições dos testes são realmente idênticas, apenas difere a orientação do robô (ou seja, a área de conforto é diferente). Como era de esperar essa orientação afeta os resultados. Ao observar-se a orientação que o robô tem no caso 2, percebe-se que era impossível o robô seguir a trajetória calculada no caso 1, pois a área de segurança iria invadir a zona de obstáculos. O mesmo se conclui ao fazer-se a comparação inversa. A trajetória gerada no caso 1 não seria válida no caso 2, pelos mesmos motivos. Em ambos os casos, o caminho ótimo foi encontrado, evitando-se colisões.

Relativamente ao manipulador, pode-se considerar que a ferramenta de trabalho ocupa uma célula, não sendo necessário criar uma área de segurança. Como dito anteriormente o planeamento de trajetórias para o manipulador é feito no mapa 3D. Contudo, devido às limitações do *rviz*,

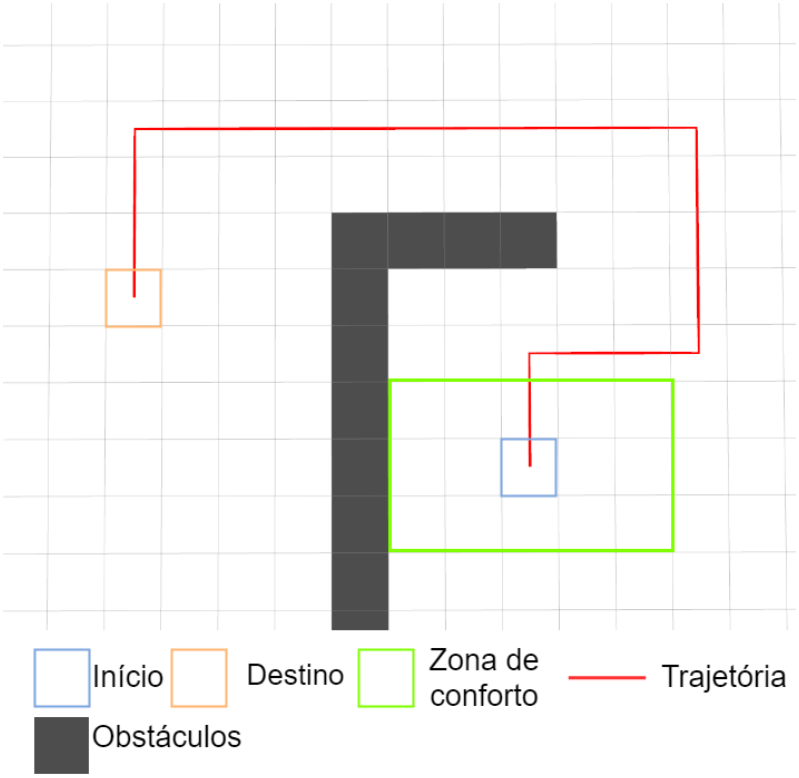


Figura 5.4: Trajetória gerada para a plataforma móvel. Caso 1.

não é possível representar um mapa 3D neste. Assim sendo, é apenas apresentada a primeira camada de z como se pode ver na figura 5.6. Devido a esta limitação, para se testar o algoritmo, colocou-se apenas obstáculos na primeira camada do mapa, deixando-se as restantes camadas livres de obstáculos. A figura 5.6 mostra os resultados obtidos. Optou-se por se colocar duas vistas diferentes, apenas para se entender melhor a movimentação da trajetória gerada no eixo z. Os resultados conseguidos vêm também validar o algoritmo implementado, uma vez que se obteve a trajetória ótima entre as células de início e destino.

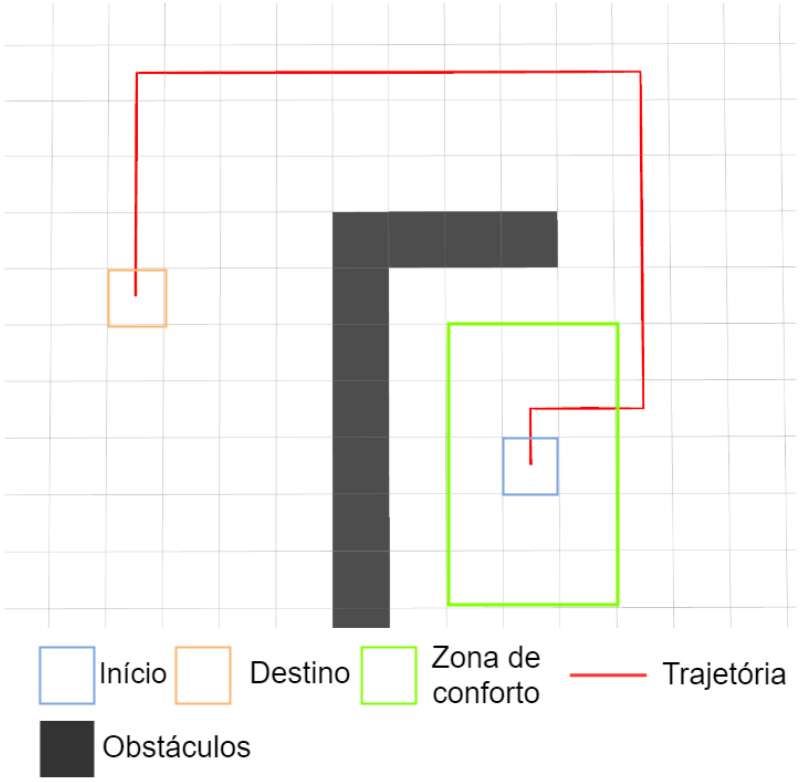


Figura 5.5: Trajetória gerada para a plataforma móvel. Caso 2.

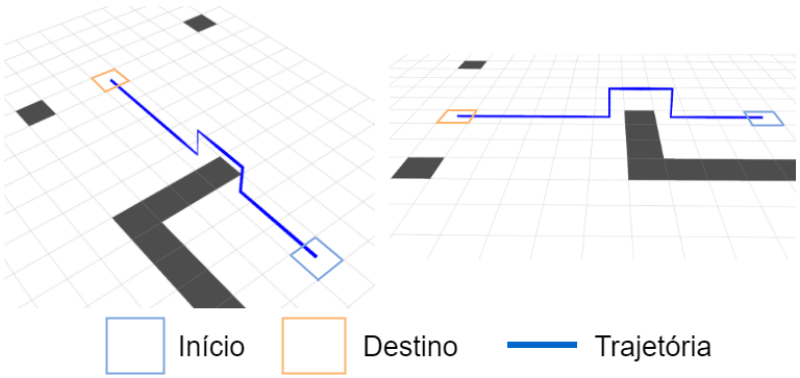


Figura 5.6: Trajetória gerada para o manipulador.

5.2 Implementação

Ao se definir um ponto inicial (P_{mi}) e final (P_{mf}) de um caminho a ser percorrido pela ferramenta de trabalho, existem infinitas configurações possíveis que alcançam esses pontos, devido ao facto do sistema ser cinematicamente redundante. Como já explicado anteriormente, a estratégia adotada passa por definir duas trajetórias, uma primeira para o manipulador e uma segunda, dependente da primeira, para a plataforma. Desta forma, um dos problemas passa por definir quais os pontos de início (P_{pi}) e fim (P_{pf}) da trajetória da plataforma. Para se definir esses pontos, pode resolver-se a cinemática inversa, recorrendo ao método explicado na secção 4.5.1, para os

pontos P_{mi} e P_{mf} do qual se obtém as configurações do sistema $q_i = [qi_{plat}\ qi_{manip}]$ e $q_f = [qf_{plat}\ qf_{manip}]$, respetivamente. Tendo estas configurações pode definir-se $P_{pi} = qi_{plat}$ e $P_{pf} = qf_{plat}$ e consequentemente gerar uma trajetória para a plataforma entre estes dois pontos.

Depois de gerada a trajetória para o manipulador, todas as células do mapa a partir das quais o manipulador não consiga alcançar a sua trajetória, são considerados obstáculos. Na figura 5.7 está representado um exemplo da zona de segurança criada em torno da trajetória do manipulador. As células pretas são os obstáculos do mapa, as cinzentas representam a zona proibida para a plataforma (classificadas desta forma, tendo em conta a trajetória azul) enquanto as células brancas são as células livres, nas quais a plataforma pode navegar.

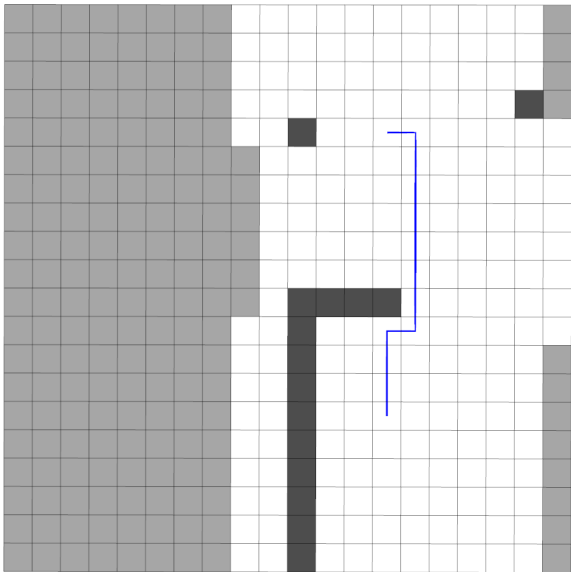


Figura 5.7: Zona de segurança em torno da trajetória do manipulador.

As trajetórias calculadas para a plataforma e para o manipulador, são constituídas por uma sequência de células que unem dois pontos. Para se fazer o seguimento desta, são seguidas duas lógicas diferentes. Relativamente ao seguimento da trajetória pela plataforma, usou-se o algoritmo 1 através do qual se faz a plataforma alcançar sequencialmente as células que formam a sua trajetória. Ou seja, move-se a plataforma para a célula inicial, quando esta for alcançada, move-se para a célula seguinte e assim sucessivamente até se chegar à célula de destino.

No que diz respeito ao seguimento da trajetória do manipulador, uma vez que o movimento da plataforma afeta a posição do manipulador e de maneira a se ter um movimento mais fluído a lógica implementada foi um pouco diferente. Colocando a ferramenta de trabalho na célula inicial, unem-se as duas primeiras células por um segmento de reta e em cada ciclo de controlo calcula-se o ponto do segmento de reta mais perto da posição da ferramenta de trabalho. De seguida, move-se a ferramenta de trabalho para esse ponto. Quando a célula seguinte for alcançada, une-se novamente essa célula e a seguinte através de um segmento de reta e repete-se o procedimento até se atingir a célula final. De seguida, será explicado o algoritmo genérico para calcular o ponto mais próximo de um segmento de reta a outro ponto (algoritmo 4).

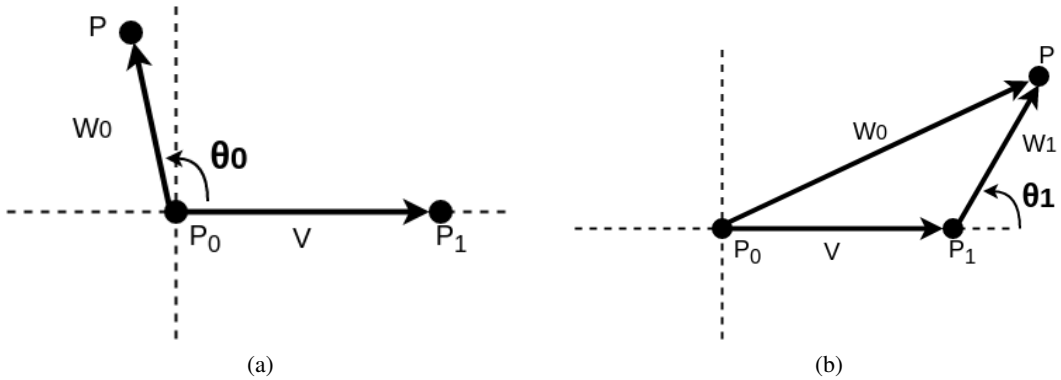


Figura 5.8: Cálculo do ponto mais perto ao ponto P.

Considerando o segmento de reta formado pelos pontos P_0 (célula atual) e P_1 (célula final), uma possibilidade para descobrir o ponto desse segmento mais perto do ponto P (posição do manipulador) é considerar o produto interno dos vários vetores envolvidos ($\vec{w_0} = P - P_0, \vec{w_1} = P - P_1$ e $\vec{v} = P_1 - P_0$) e testar se são positivos, negativos ou zero. Existem 3 possibilidades: o ponto mais próximo à reta é P_0 ou é P_1 ou então está entre estes dois pontos. Fazendo o produto interno $w_0.v$, caso este seja menor que 0, significa que o ângulo entre os dois vetores é maior que 90º, logo o ponto mais próximo de P é P_0 (figura 5.8a). Caso a condição anterior não se verifique e se $w_1.v > 0$, então $\theta_1 < 90^o$ e consequentemente o ponto mais próximo de P é P_1. Esta condição é equivalente a $w_0.v > v.v$, pois $w_0 = v + w_1$ (5.8b). Caso nenhuma destas condições se verifique, faz-se então a projeção de P no segmento de reta formado por P_0 e P_1 e calcula-se a distância desse ponto a P_0. Basta depois multiplicar essa distância pelo vetor de direção $\vec{v}$.

Algoritmo 4: Closest Point

input : P,P_1 e P_2
output: Ponto do segmento de reta mais perto de P

```
1  v = P1 − P0;
2  w = P − P0;
3  c1 = w · v;
4  c2 = v · v;
5  if c1 < 0 then
6  |  return P0;
7  end
8  if c2 < c1 then
9  |  return P1;
10 end
11 b = c1/c2;
12 Pb = P0 + bv;
13 return Pb;
```

5.2.1 Algoritmo implementado

De forma a se entender melhor a lógica explicada anteriormente para o seguimento de ambas as trajetórias, recorreu-se a um diagrama de atividade 5.9.

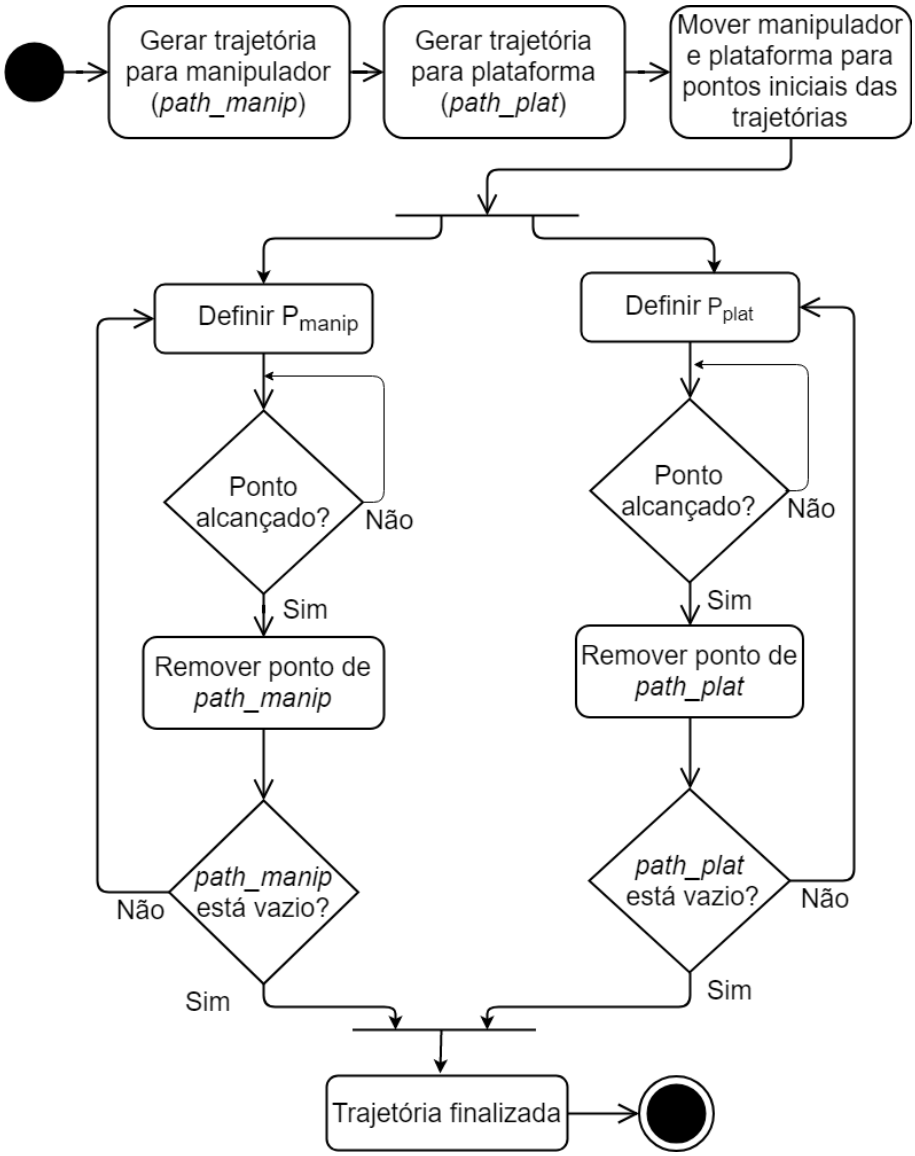


Figura 5.9: Diagrama de atividade para seguimento das trajetórias.

Relativamente à terminologia utilizada, *path_manip* e *path_plat* guardam as células que formam cada trajetória. P_{manip} e P_{plat} representam o próximo ponto de destino do manipulador e da plataforma, respetivamente. Como explicado anteriormente, P_{plat} é igual à próxima célula presente em *path_plat* enquanto P_{manip} é obtido recorrendo ao algoritmo 4.

À medida que os pontos de destino vão sendo alcançados pela plataforma e pelo manipulador, vão sendo definidos novos pontos. O algoritmo termina quando ambos as trajetórias forem cumpridas.

5.2.2 Integração ROS

Para integrar o planeador de trajetórias com o restante sistema, desenvolveu-se um nó *ROS* responsável por correr o algoritmo A*. Esse nó subscreve vários tópicos que lhe dão informação do mapa, da célula inicial e da célula de destino ($/astar/map$, $/astar/initial\_position$, $/astar/position\_goal$, respetivamente). Sempre que novas posições de início e destino são publicadas, calcula-se a nova trajetória e publica-se o resultado obtido no tópico $/astar/Ur5Path$ ou $/astar/OmniPath$, caso a trajetória seja referente ao manipulador ou à plataforma, respetivamente. Na figura 5.10 pode ver-se a arquitetura *ROS* utilizada.

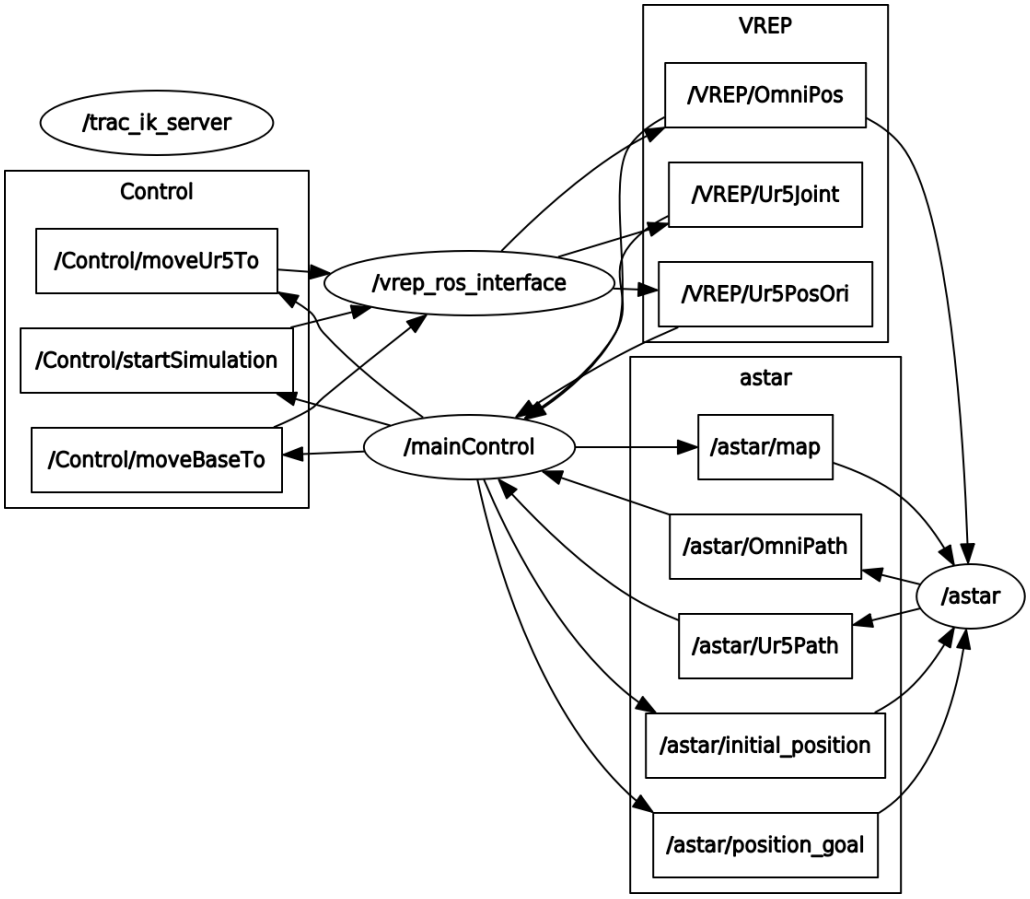


Figura 5.10: Arquitetura *ROS*.

5.3 Resultados

Como forma de teste, desenvolveu-se um ambiente de simulação no *V-REP* com alguns obstáculos (figura 5.11). Definiram-se ainda dois pontos de início e destino.

Inicialmente optou-se por um problema mais simples, no sentido em que não se colocou obstáculos ao nível do manipulador entre os pontos definidos. Desta forma, espera-se que a trajetória gerada para o manipulador seja sempre em linha reta, enquanto a trajetória da plataforma obrigará a desvios de obstáculos. A figura 5.11 mostra o resultado do ambiente criado.

Na figura 5.12 pode ver-se os resultados obtidos para as trajetórias do manipulador e da plataforma. Analisando-o, percebe-se que os resultados então dentro do esperado. A trajetória azul

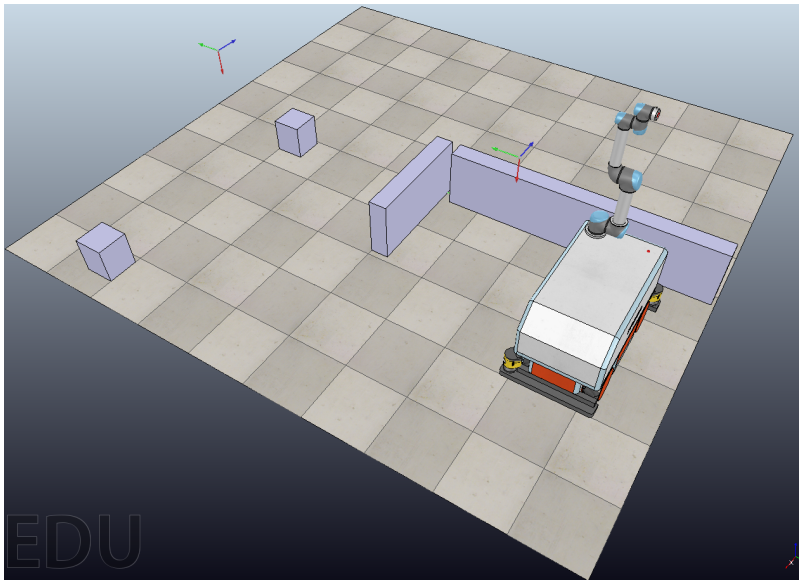


Figura 5.11: Ambiente de simulação.

(referente ao manipulador) é uma reta entre os pontos de início e fim, enquanto a trajetória vermelha (correspondente à plataforma) efetua desvios para evitar colisões com obstáculos.

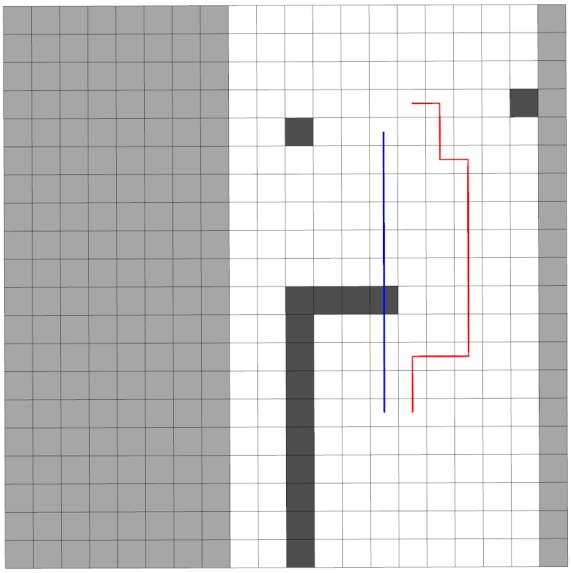


Figura 5.12: Trajetórias resultantes do A*. A azul a trajetória do manipulador e a vermelho a trajetória da plataforma.

Para se entender melhor os resultados obtidos no simulador *V-REP*, marcou-se na simulação com a cor vermelha, a trajetória a ser seguida pelo manipulador. Na figura 5.13 pode ver-se alguns *screenshots* tirados ao longo da trajetória. Pela análise dos mesmos, constata-se que a ferramenta de trabalho acompanhou sempre a linha vermelha, mesmo quando o movimento da plataforma

mudava de direção. Desta forma, pode concluir-se que a trajetória do manipulador foi efetuada com sucesso. Relativamente à plataforma, esta evitou os obstáculos ao longo de toda a trajetória, além disso, confirma-se que a trajetória gerada para a plataforma permite ao manipulador cumprir a sua própria trajetória. Outro fator relevante, prende-se com a zona de segurança criada em torno da célula que representa a plataforma. Como se pode ver na figura 5.12, a trajetória da plataforma

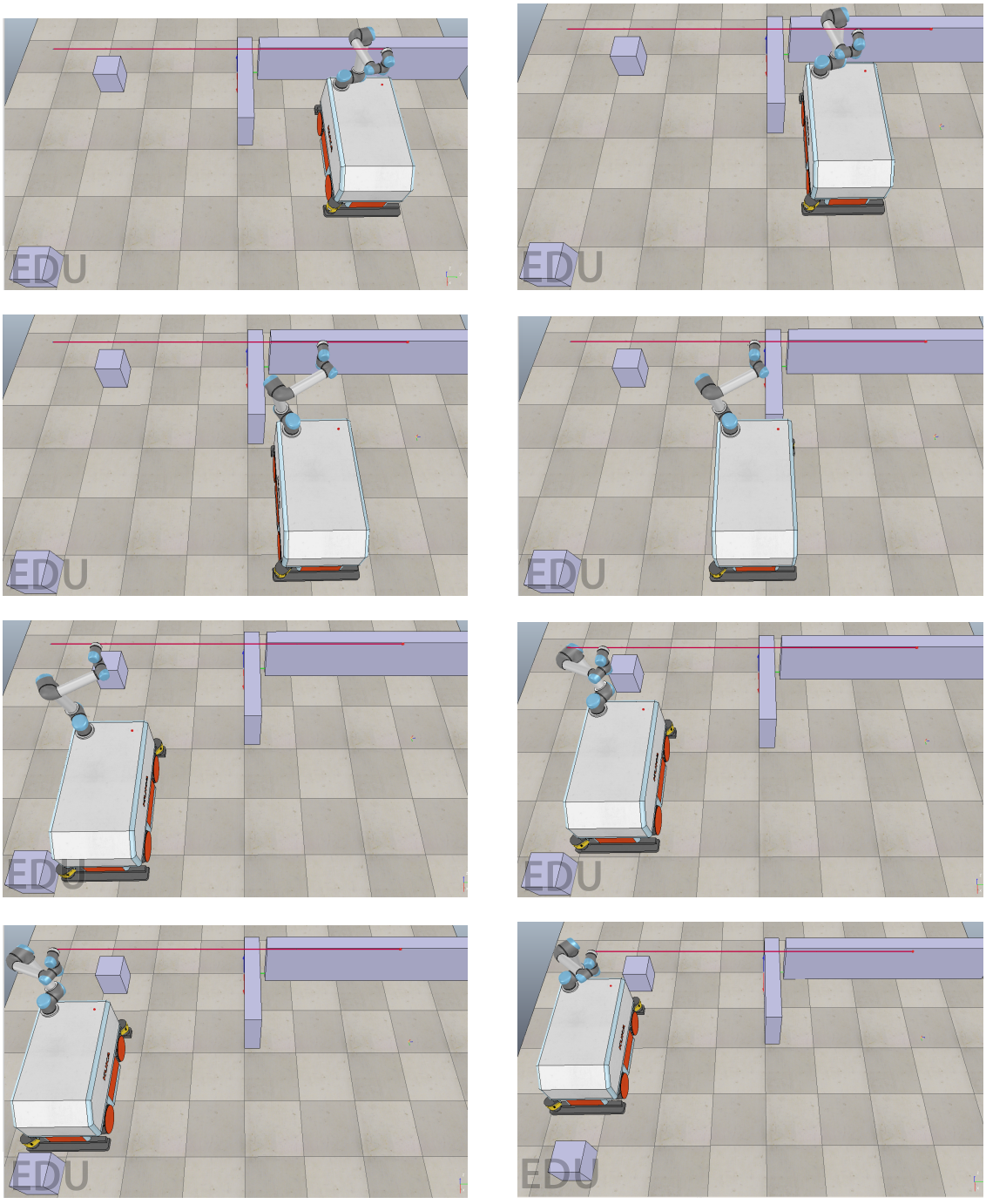


Figura 5.13: Resultados do primeiro teste no simulador *V-REP*.

passa "longe" dos obstáculos, contudo os *screenshots* mostram a plataforma a passar perto dos obstáculos, o que demonstra que a zona de segurança foi bem delimitada.

Para uma análise mais profunda dos resultados obtidos, é necessário complementar a informação visual que se retirou da figura 5.13. Para tal, ao longo do teste foram sendo guardadas as posições da plataforma e do manipulador, recorrendo à ferramenta *Rosbag*. Deste modo, é possível comparar-se estes valores com os valores pretendidos, ou seja, comparar-se com as trajetórias pretendidas. Os resultados obtidos podem ser analisados na figura 5.14.

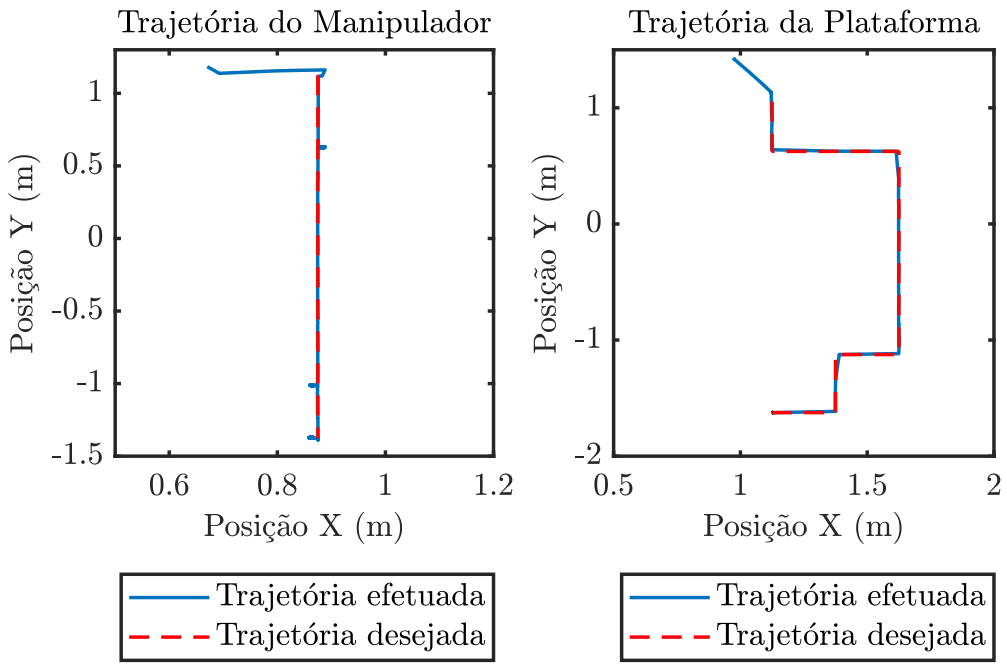


Figura 5.14: Comparação das trajetórias desejadas e efetuadas.

É possível ver-se na parte superior de ambos os gráficos que as trajetórias efetuadas não estão de acordo com as suas trajetórias pretendidas. Isto deve-se ao facto de que no início da simulação, tanto o manipulador como a plataforma não estão nas posições iniciais das suas trajetórias, ou seja, essa parte de ambos os gráficos representam a movimentação da plataforma e do manipulador para as posições iniciais das duas trajetórias.

Depois de alcançadas essas posições, inicia-se então o seguimento das trajetórias. Os resultados obtidos são satisfatórios, uma vez que, tanto o manipulador como a plataforma seguiram as trajetórias que eram pretendidas. Era ainda de esperar que estas trajetórias fossem parecidas com as obtidas na figura 5.12. Comparando as duas figuras, conclui-se que, a menos de um fator de escala, as trajetórias são iguais.

Posteriormente, efetuou-se um segundo teste no qual se alterou um pouco o ambiente de simulação, colocando-se obstáculos ao nível do manipulador como se pode ver na figura 5.15. Os pontos de início e destino mantiveram-se iguais. Este ambiente de simulação torna o seguimento da trajetória mais difícil, uma vez que tanto o manipulador como a plataforma são obrigados a desviarem-se de obstáculos.

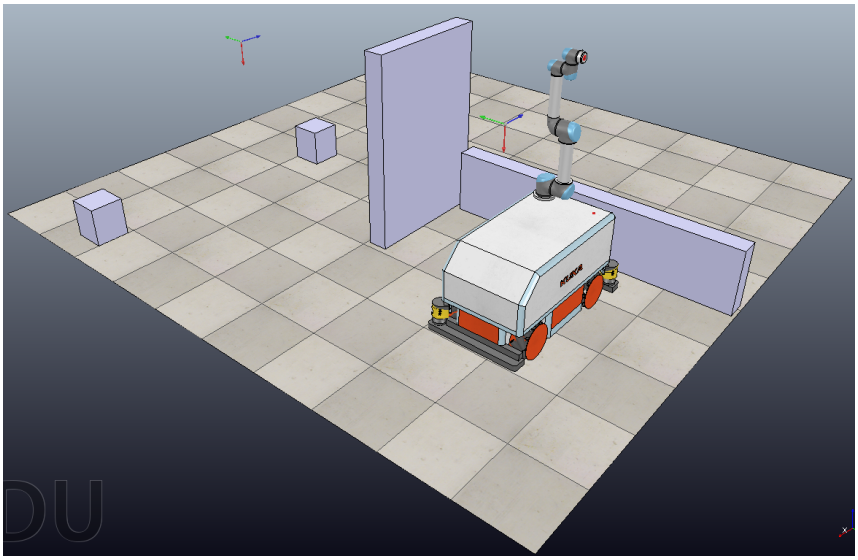


Figura 5.15: Novo ambiente de simulação.

Na figura 5.16 observa-se as trajetórias obtidas neste novo ambiente de simulação. Como seria de esperar, a trajetória do manipulador é diferente da obtida na simulação anterior devido aos obstáculos existentes entre o ponto inicial e de destino. Relativamente à trajetória da plataforma, os pontos de início e destino calculados também são diferentes do obtidos no teste anterior, o que faz com que a trajetória seja diferente.

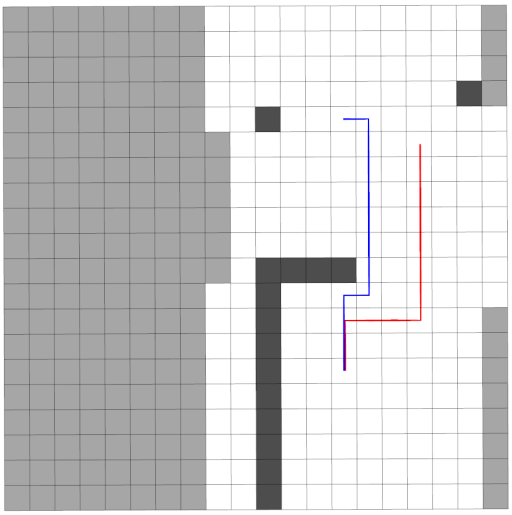


Figura 5.16: Trajetórias resultantes do A* para o novo ambiente de simulação. A azul a trajetória do manipulador e a vermelho a trajetória da plataforma.

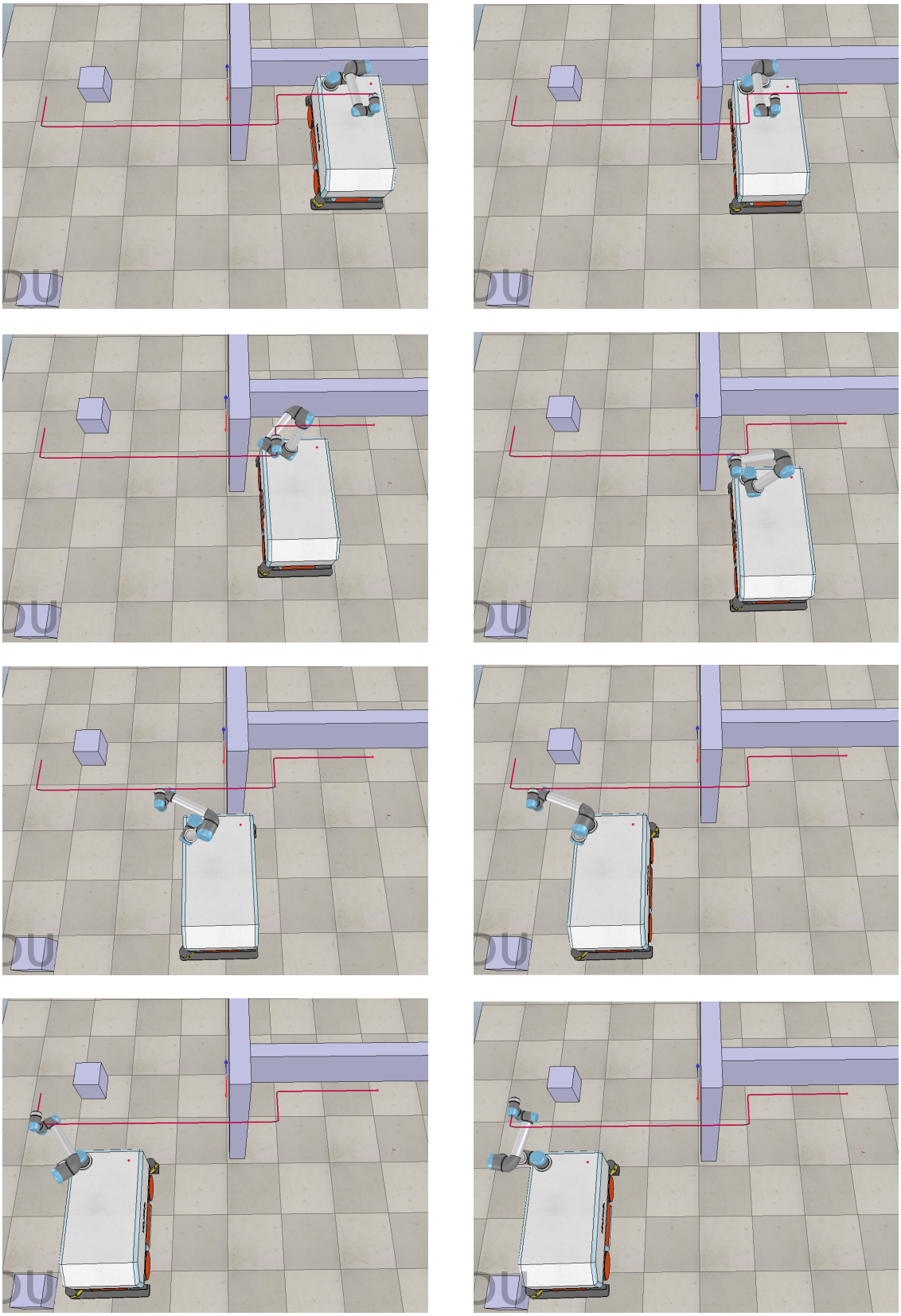


Figura 5.17: Resultados do segundo teste no simulador *V-REP*.

Para se analisar os resultados, marcou-se novamente no ambiente de simulação a trajetória gerada para o manipulador. Na figura 5.17 pode ver-se alguns *screenshots* tirados ao longo da simulação. A análise destes mostram a ferramenta de trabalho a acompanhar a sua trajetória durante toda a simulação, contudo, a análise da figura 5.18 demonstra que o manipulador em certo ponto se desviou ligeiramente da sua trajetória. Posteriormente a este desvio, a ferramenta continuou a percorrer a sua trajetória corretamente.

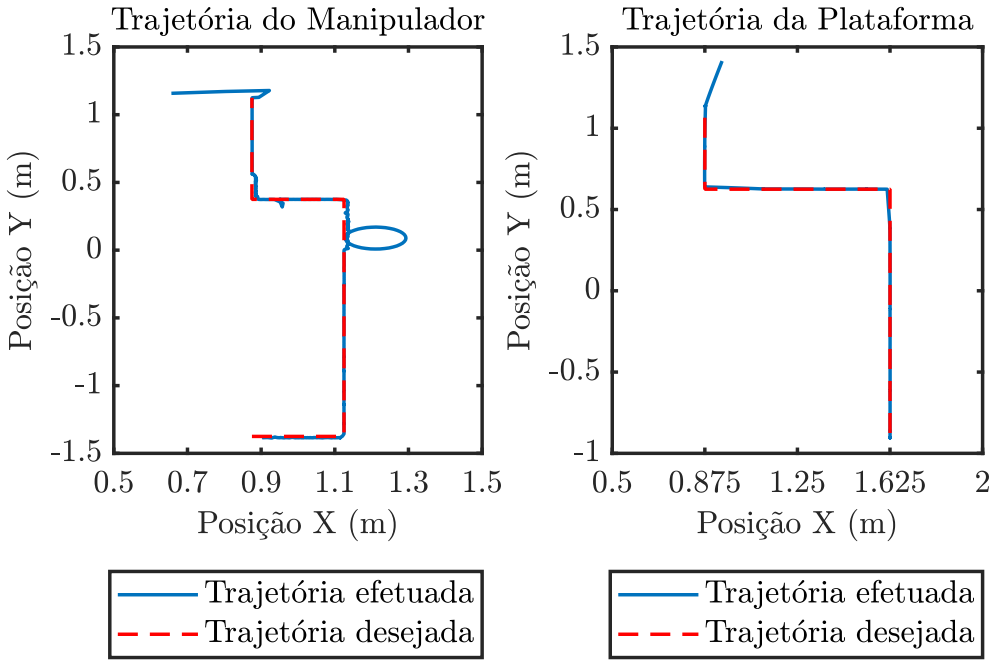


Figura 5.18: Comparação das trajetórias desejadas e efetuadas.

Apesar das juntas rotativas presentes no manipulador UR5 terem uma capacidade de rotação de 360º, estas possuem limites físicos que as impedem de girar continuamente sobre o seu eixo. No que respeita a esses limites, os mesmos estão definidos num intervalo de $-\pi$ a π. Desta forma, quando alguma junta trabalha perto do limite deste intervalo existe a possibilidade de movimentos indesejados.

Na figura 5.19 está representado o círculo trigonométrico, o limite das juntas rotativas, e as orientações A e B. Tomando esta figura como exemplo, caso a junta rotativa esteja com a orientação A e se queira mudar para a orientação B, o caminho que minimiza a distância entra as duas posições é obtido rodando no sentido horário. Contudo, devido à existência do limite é impossível a junta fazer esse trajeto, assim sendo é necessário rodar no sentido anti-horário. São este tipo de rotações que criam movimentos indesejados e fazem o manipulador afastar-se da sua trajetória.

Para se diminuir o impacto desta transição no seguimento da trajetória é feita uma supervisão das coordenadas das juntas que se enviam para o simulador. Comparando o valor dessas coordenadas com as coordenadas atuais do manipulador é possível detetar se irá existir uma transição

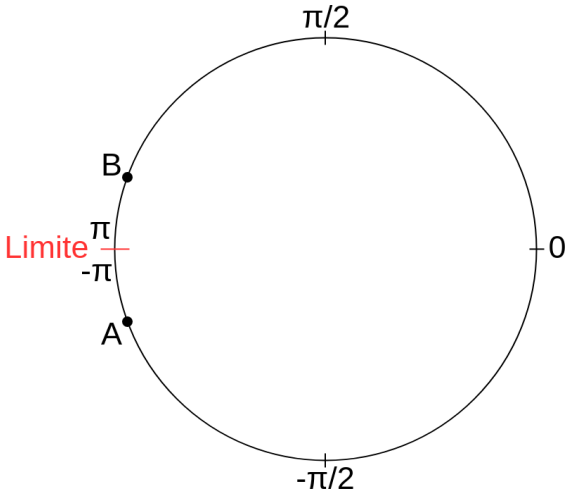


Figura 5.19: Representação do limite das juntas rotativas no círculo trigonométrico.

em algumas das juntas. Caso se verifique esta situação, então o mais seguro é parar a plataforma enquanto o manipulador altera as suas coordenadas para as coordenadas desejadas. Além disso, caso o manipulador esteja a executar alguma tarefa, como por exemplo, pintura, ao se detetar que vai existir uma transição, é possível interromper essa mesma tarefa. Depois de o manipulador se colocar na posição desejada, a plataforma pode continuar o seguimento da sua trajetória, assim como o manipulador pode recomeçar a tarefa que estava a executar anteriormente.

Desta forma, o limite das juntas foi responsável pelo movimento indesejado que se vê na figura 5.18. Analisando o comportamento das juntas ao longo da simulação (figura 5.20), observa-se que a junta 5 do manipulador se aproxima da posição π, ou seja, aproxima-se do seu limite, movendo-se depois para um valor perto de $-\pi$. Esta transição na posição traduz-se no movimento que se observou anteriormente. A restante trajetória foi cumprida com sucesso por parte do manipulador. O mesmo se pode concluir relativamente à plataforma.

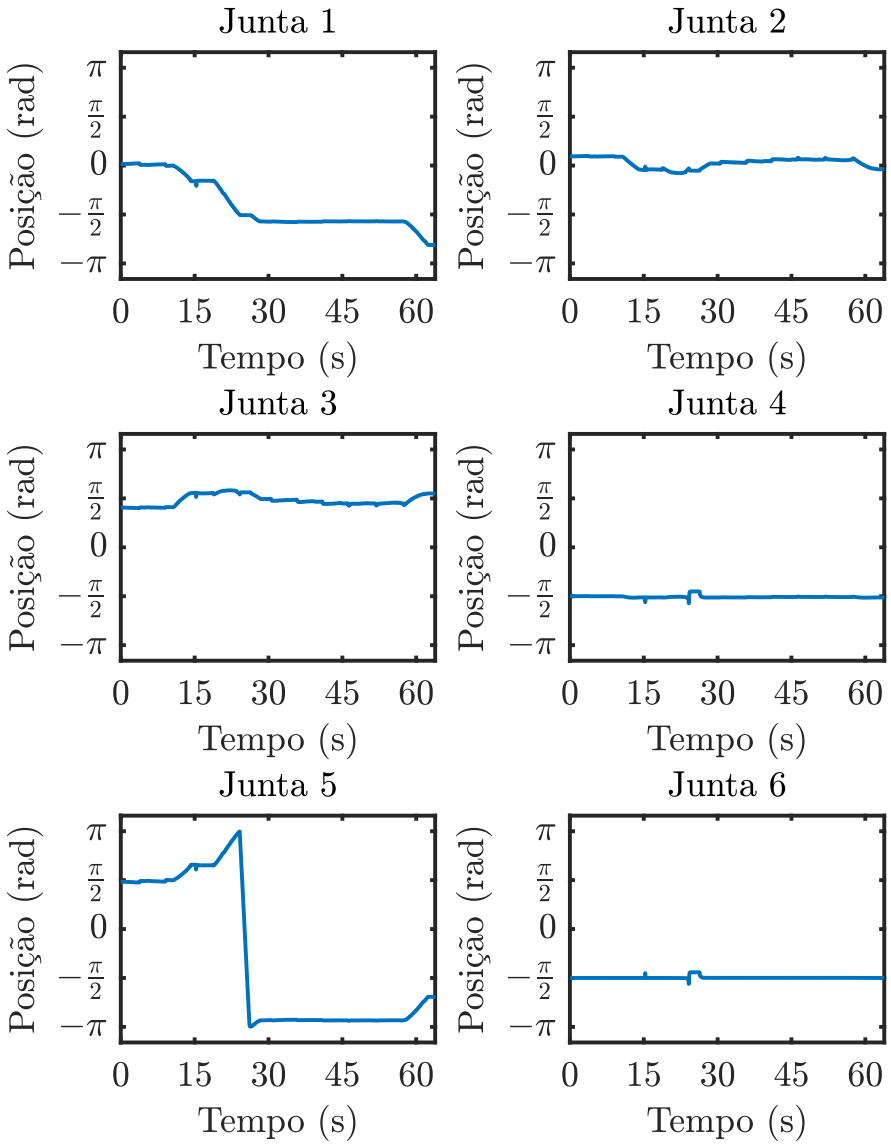


Figura 5.20: Evolução das coordenadas das juntas ao longo da simulação.

5.4 Conclusão

Este capítulo lidou com o problema de planeamento de trajetórias, utilizando o algoritmo A* para encontrar o caminho mínimo entre dois pontos. Para garantir que o manipulador consegue viajar do ponto A até ao ponto B evitando colisões, seguiu-se uma abordagem na qual se calcula uma trajetória para o manipulador, posteriormente com base nessa trajetória gera-se uma outra para a plataforma móvel. Esta abordagem permitiu ao manipulador seguir a sua trajetória, evitando colisões tanto do manipulador como da plataforma móvel com os obstáculos presentes no mapa. Foi também possível verificar que é importante garantir que as juntas não trabalhem perto do

seu limite. Caso isso aconteça, verifica-se que existem movimentos indesejados por parte do manipulador.

Capítulo 6

Conclusões e propostas de trabalho futuro

O objetivo desta dissertação passou por estudar e desenvolver a cinemática composta de um manipulador móvel, constituído pelo manipulador UR5 e por um robô omnidirecional de 4 rodas. Foi também desenvolvido um planeador de trajetórias para se calcular o caminho ótimo entre dois pontos, evitando obstáculos. Para se implementar e validar as soluções desenvolvidas ao longo da dissertação, usou-se a ferramenta *ROS* e o simulador *V-REP* de forma a testar a navegação do manipulador móvel. Será agora apresentada uma síntese do trabalho realizado e as suas principais conclusões.

Inicialmente começou-se por analisar a cinemática do manipulador e da plataforma móvel separadamente. Relativamente ao manipulador, ao se estudar a sua cinemática consegue-se perceber a interferência das juntas na posição da ferramenta de trabalho. Além disso, através da sua cinemática é possível calcular a sua manipulabilidade. Quanto à plataforma móvel, o estudo da sua cinemática é essencial para se fazer o controlo do mesmo, uma vez que permite relacionar a velocidade da plataforma com a velocidade de cada roda.

A primeira abordagem para resolução do problema foi baseada num controlo do manipulador e plataforma móvel de forma independente. Desta forma, caso o ponto esteja fora do alcance do manipulador então a plataforma aproxima-se do ponto. Esta solução provou ser simples, contudo não é ideal caso se queira maximizar a manipulabilidade do manipulador. Além disso, caso se queira alcançar vários pontos sucessivamente, o movimento realizado não é o ideal.

Foi então proposta uma solução na qual o movimento da plataforma é aproximado pelo movimento de um manipulador de 3 juntas. As primeiras duas juntas, do tipo prismáticas, representam o movimento da plataforma no plano xy enquanto a terceira, do tipo rotativa, representa a rotação da plataforma em torno de z. Ao anexar este manipulador ao manipulador UR5, obtém-se um manipulador de 9 juntas. A cinemática deste manipulador modela a cinemática do sistema "plataforma omnidirecional + UR5". Para resolver a sua cinemática, devido à sua complexidade, foi necessário recorrer à biblioteca *trac_ik*, na qual se fez algumas alterações necessárias. Esta solução apresentou bons resultados, uma vez que tratar o sistema como um todo permite garantir

restrições (como por exemplo, máxima manipulabilidade) de uma forma que não seria possível caso se resolvesse a cinemática da plataforma e do UR5 de forma separada.

Relativamente ao planeamento de trajetórias, foi desenvolvido o algoritmo A*. Este apresentou bons resultados a gerar trajetórias tanto em ambiente 2D como em ambiente 3D. Depois de obtida a trajetória do manipulador, visto que resolução da cinemática do sistema integrado não garante que a plataforma móvel não colida com obstáculos, foi então necessário propor outra solução para o problema. A abordagem proposta passou por definir duas trajetórias distintas, uma para o manipulador e outra para a plataforma. Relativamente à trajetória do manipulador, definidos um ponto de início e destino, calculou-se o caminho ótimo entre esses pontos através do A*. Com base nessa trajetória, calculou-se uma trajetória para a plataforma que garante que o manipulador consegue cumprir a sua. Esta abordagem mostrou bons resultados, na medida em que o manipulador conseguiu cumprir as trajetórias definidas, evitando obstáculos. Contudo, quando alguma das juntas que constituem o manipulador UR5 trabalha perto do seu limite físico, podem surgir movimentos imprevisíveis.

6.1 Trabalho futuro

Visando a melhoria do trabalho aqui desenvolvido, como trabalho futuro seria interessante alterar-se a biblioteca *trac-ik* de forma a se conseguir mudar dinamicamente os limites das juntas 1 e 2 do sistema (responsáveis pelo posicionamento da plataforma em xy). Desta forma, seria possível restringir a solução da cinemática inversa, de maneira a garantir que a plataforma móvel se posicionava dentro de uma área pretendida. Esta melhoria permitiria simplificar a questão do planeamento de trajetórias, pois depois obtida a trajetória para o manipulador, não seria necessário gerar uma trajetória para a plataforma, bastando apenas garantir que a solução da cinemática inversa não posiciona a plataforma móvel numa zona ocupada por obstáculos.

Seria ainda importante garantir que ao longo de uma trajetória, as juntas do UR5 não trabalhem perto do seu limite físico, de forma a evitar transições na zona de limite. Tal poderia ser feito, escolhendo a configuração do UR5 que mais se adequa à trajetória em questão.

Bibliografia

[1] *Executive Summary World Robotics 2017 Industrial Robots* [Online]. Available: https://ifr.org/downloads/press/Executive_Summary_WR_2017_Industrial_Robots.pdf.

[2] J. Wallén, *The history of the industrial robot*. Linköping University Electronic Press, 2008.

[3] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, "Autonomous mobile robots," *Massachusetts Institute of Technology*, 2004.

[4] *Home - Stamina* [Online]. Available: http://stamina-robot.eu/.

[5] *Colrobot* [Online]. Available: https://www.colrobot.eu/.

[6] *ScalABLE 4.0 – Development and demonstration of an (OSPS)* [Online]. Available: https://www.scalable40.eu/.

[7] *ILIAD Project – An EU-funded research project on on Intra-Logistics with Integrated Automatic Deployment for safe and scalable fleets in shared spaces* [Online]. Available: http://iliad-project.eu/.

[8] *Ridgeback - Omnidirectional mobile manipulation robot* [Online]. Available: https://www.clearpathrobotics.com/ridgeback-indoor-robot-platform/.

[9] J. N. Pires, *Industrial robots programming: building applications for the factories of the future*. Springer Science & Business Media, 2007.

[10] P. V. G. Simplício and B. R. Lima, "Manipuladores robóticos industriais," *Caderno de Graduação-Ciências Exatas e Tecnológicas-UNIT*, vol. 3, no. 3, p. 85, 2016.

[11] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot modeling and control*, vol. 3. Wiley New York, 2006.

[12] E. Ostergaard, "Lightweight robot for everybody [industrial activities]," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 17–18, 2012.

[13] *Estudos de casos da Universal Robots | Robôs colaborativos operam em empresas de todo o mundo* [Online]. Available: https://www.universal-robots.com/pt/hist%C3%B3rias-de-casos/.

[14] *UR5 - The flexible and collaborative robotic arm* [Online]. Available: https://www.universal-robots.com/pt/produtos/ur5/.

[15] *Universal Robots - Detalhes técnicos* [Online]. Available: https://www.universal-robots.com/media/1514588/101081_199922_ur5_technical_details_web_a4_art03_rls_por.pdf.

[16] S. G. Tzafestas, "Mobile robots-1: General concepts," 2014.

[17] H. Taheri, B. Qiao, and N. Ghaeminezhad, "Kinematic model of a four mecanum wheeled mobile robot," *International Journal of Computer Applications*, vol. 113, no. 3, 2015.

[18] P. L. C. G. d. Costa *et al.*, "Planeamento cooperativo de tarefas e trajectórias em múltiplos robôs," 2011.

[19] J. Pearl, "Heuristics: Intelligent search strategies for computer problem solving," 1984.

[20] M. Likhachev, G. J. Gordon, and S. Thrun, "Ara*: Anytime a* with provable bounds on sub-optimality," in *Advances in neural information processing systems*, pp. 767–774, 2004.

[21] P. P. Chakrabarti, S. Ghose, A. Acharya, and S. De Sarkar, "Heuristic search in restricted memory," *Artificial intelligence*, vol. 41, no. 2, pp. 197–221, 1989.

[22] R. Zhou and E. A. Hansen, "Memory-bounded a* graph search," in *Proceedings of the Fifteenth International Florida Artificial Intelligence Research Society Conference*, pp. 203–209, AAAI Press, 2002.

[23] A. Stentz, "Optimal and efficient path planning for partially-known environments," in *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, pp. 3310–3317, IEEE, 1994.

[24] S. Koenig and M. Likhachev, "Fast replanning for navigation in unknown terrain," *IEEE Transactions on Robotics*, vol. 21, no. 3, pp. 354–363, 2005.

[25] Y. Yamamoto and X. Yun, "Coordinating locomotion and manipulation of a mobile manipulator," in *Decision and Control, 1992., Proceedings of the 31st IEEE Conference on*, pp. 2643–2648, IEEE, 1992.

[26] B. B. Parodi, *Controle coordenado de posição e força de um manipulador sobre plataforma móvel emulando um rov*. PhD thesis, Universidade Federal do Rio de Janeiro, 2003.

[27] T. Yoshikawa, "Translational and rotational manipulability of robotic manipulators," in *American Control Conference, 1990*, pp. 228–233, IEEE, 1990.

[28] M. J. Tsai and Y. H. Chiou, "Manipulability of manipulators," *Mechanism and Machine Theory*, vol. 25, no. 5, pp. 575–585, 1990.

[29] T. T. Phan, T. L. Chung, M. D. Ngo, H. K. Kim, and S. B. Kim, "Decentralized control design for welding mobile manipulator," *Journal of mechanical science and technology*, vol. 19, no. 3, pp. 756–767, 2005.

[30] N. Egerstedt and X. Hu, "Coordinated trajectory following for mobile manipulation," in *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, vol. 4, pp. 3479–3484, IEEE, 2000.

[31] J. Denavit and R. Hartenberg, "A kinematic notation for lower pair mechanisms based on matrices," pp. 215–221, 1955. cited By 8.

[32] R. Keating and J. H. University, "Ur5 inverse kinematics,"

[33] Y. Xiao, Z. Fan, W. Li, S. Chen, L. Zhao, and H. Xie, "A manipulator design optimization based on constrained multi-objective evolutionary algorithms," in *Industrial Informatics-Computing Technology, Intelligent Technology, Industrial Information Integration (ICIICII), 2016 International Conference on*, pp. 199–205, IEEE, 2016.

[34] P. Beeson and B. Ames, "Trac-ik: An open-source library for improved solving of generic inverse kinematics," in *Humanoid Robots (Humanoids), 2015 IEEE-RAS 15th International Conference on*, pp. 928–935, IEEE, 2015.